VISUAL SEARCH INTERFACES FOR ONLINE DIGITAL REPOSITORIES

A Dissertation
Presented to
The Academic Faculty

By

Edward C. Clarkson

In Partial Fulfillment
Of the Requirements for the Degree
Doctor of Philosophy in Computer Science

Georgia Institute of Technology

August 2009

VISUAL SEARCH INTERFACES FOR ONLINE DIGITAL REPOSITORIES

Approved by:

Dr. James D. Foley, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Colin Potts
School of Interactive Computing
Georgia Institute of Technology

Dr. Gregory Abowd
School of Interactive Computing
Georgia Institute of Technology

Dr. John Stasko
School of Interactive Computing
Georgia Institute of Technology

Dr. Gary Marchionini
School of Library and Information Science
University of North Carolina at Chapel Hill

Date Approved: May 16, 2009

For Molley

ACKNOWLEDGEMENTS

I would like to think that this section is unnecessary because I have made my gratitude known everyone who has helped me finish this dissertation. But so often the proverbial thought-that-counts goes unmentioned; I am happy for this chance to give my overt thanks. This is especially true for family, whose support we so often take for granted. That I take my wife Molley for granted is a measure of her importance and comfort to me: life together with her and Libby (and those to come, God willing) is the only one I can imagine. I thank my parents for a wonderful childhood: I never got a Nintendo but I never lacked a computer or the expectation to do my best. My sister Evie deserves recognition for putting up with my shenanigans (from heckling to Monopoly). I have so many wonderful grandparents, aunts, uncles, cousins and in-laws; I feel enormously lucky to have an extended family made of people I look forward to seeing and would choose as friends were we not related.

Professionally, I am indebted to my advisor Jim Foley for his patient counsel and firm motivation, each in its due measure. I also thank my committee—Gregory Abowd, Gary Marchionini, Colin Potts and John Stasko—for their help in completing this dissertation. I am glad Jason Day, my only lab-mate, was a good one, and I hope he enjoyed our nonsense as much as I did. Collaboration with other faculty has been one of my favorite aspects of Georgia Tech, and I have enjoyed the opportunity to work with Ron Arkin, Sham Navathe, Thad Starner and Jeff Pierce over the years. But it is students who really make research go, and I have been equally lucky to work with excellent current and former students: James Clawson, Krishna Desai, Kent Lyons and Shwetak

Patel. On specific projects, the HCC EDL would not have been possible without the work of Andy Cox, Andy Wu and Remco Groeneweg. I also think my collaborators in the College of Architecture—Profs. Sabir Khan and Benjamin Flowers and Ph.D. students Myung Seok Hyun, Carina Antunez and Marietta Monaghan—for their generous assistance with deploying my research systems to their classes.

Universities are more pleasant places to work when there is staff who know how to get things done: it is a testament to the entire SIC and GVU staff that students consider them authorities of the first rather than the last resort. My special thanks go to Randy Carpenter, Chrissy Hendricks, Brian Poole, Charmion Richards and Don Schoner for their help in keeping little things from becoming big ones.

The Ph.D. experience would not be what it is without fellow students. Nick Diakopoulos has been a valued sounding board since our qualifying exam. Aside from being my oldest friend, Abe Crystal has the inestimable honor of both seeding the idea for the ResultMap concept (and the name). I truly appreciate the generosity of Beki Grinter and her students in the Work 2 Play lab—Marshini Chetty, Andrea Grimes, Ja-Young Sung and Susan Wyche—for sharing freely their time, equipment and friendship.

Finally, I would like to thank Stephen Fleming and the Georgia Tech College of Computing Stephen Fleming Chair in Telecommunications, and the National Visualization and Analytics Center (NVAC™)—a U.S. Department of Homeland Security Program sponsoring the Southeast Regional Visualization and Analytics Center—for financial support of my work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This work presents our research into visualization for digital repository search interfaces, motivated by the prevalence of existing hierarchical data structures and the general lack of contextualization present in existing systems. We develop the ResultMap concept, a treemap-based visualization system that we have applied to keyword search engine and faceted classification data environments, and present the results of their empirical evaluation.

We organize this work as follows: Chapter 1 provides an introduction to our problem area, motivates our general approach of leveraging hierarchical structure (via ResultMaps) for context, and proposes a thesis statement and corresponding research questions. Chapter 2 discusses related work, and includes a survey and design characterization of faceted navigation tools. Chapter 3 defines the key visual and interactive features of the ResultMap concept and justifies their basic design.

Chapter 4 presents our implementation and evaluation of ResultMaps applied to digital library search engine result pages (SERPs). Chapter 5 consists of two major portions: a presentation of our formal data and query models for faceted environments and our implementation and evaluation of ResultMaps in a faceted UI context. In Chapter 6 we conclude—based on our results from Chapter 4 and Chapter 5—with a set of principles for designing both visual search interfaces themselves and designing their evaluation. We finish with suggestions for future research in this area.

CHAPTER 1:

INTRODUCTION

Hierarchy is a fundamental organizational paradigm. People use hierarchy to abstract key concepts from groups of similar items and to help structure their thinking and reasoning. Hierarchies have long been a subject of research in computing. One common use of hierarchy is as an ontology, such as the Library of Congress classification scheme[1] or the ACM Computing Classification system[2]. As libraries have moved onto the internet and World Wide Web (WWW), so has their use of hierarchical classifications. To wit: central components of our own digital libraries have been the hierarchical topic classifications we have developed for their specific fields (Human-Centered Computing and Visual Analytics) [24].

Naturally, such hierarchies are often a central component for browsing online repositories, including ours. But their comparable use in repository search is generally limited. Conventional search relevancy reports allow users to limit searches to certain sections of a hierarchy and perhaps return sections of a hierarchy as responses to a search query. But beyond these uses, hierarchies are often underutilized in the context of repository search and exploration.

Another salient topic is *faceted classification* (FC), which categorizes items in a collection around multiple semantically orthogonal characteristics (or *facets*)[3]. The result of that process is known as *faceted metadata*, and systems based on this concept employ

[1] http://www.loc.gov/catdir/cpso/lcco/
[2] http://www.acm.org/class/1998/
[3] The actual FC process of dividing metadata between simple attributes or facets is complex and outside the scope of this work.

faceted navigation. Facets have recently come to influence research and systems in computing after a much longer history in library science. Documents in a faceted system can have any number of faceted attributes that are each either flat or hierarchical metadata. For example, a set of professional baseball players could have a facet for a player's team, his position, the length of his career, etc. Any or all of these facets might be hierarchical as well (e.g., *League* > *Division* > *Team*). Contrast facets with traditional single-attribute classifications like the Dewey Decimal System or ACM classification system mentioned above, which organize documents around a single hierarchical scheme.

We initially developed *ResultMaps*, a treemap-based query visualization system for hierarchical metadata, with simple hierarchies in mind [26]. Treemaps are a well-known tree layout method in information visualization (infovis) for displaying arbitrary tree structures [52]. They recursively divide an area into smaller sub-regions, placing nodes within their parent regions until the entire area is filled. ResultMaps use hierarchical structure (e.g., a subject classification) to map each repository document into a treemap and highlight items that correspond to query results. The hierarchical structure comes from some metadata field, which we refer to as the *mapped attribute* or *mapped variable*. Note that single-hierarchy classifications are also simply degenerate faceted schemes. As such, we apply ResultMaps both to the search engine results page (SERP) of a traditional query string engine and to the more general case of faceted metadata, with one ResultMap per hierarchical facet.

ResultMaps encode the full contents of a hierarchy, accentuate certain nodes as indicated by a query engine (e.g., by relevance to query terms) and interactively link those nodes to traditional text listings. The text listing leverages users' familiarity with

that environment and also provides them a connection to a more sophisticated ResultMap representation of the same items. Encoding an entire hierarchical structure preserves consistency between queries, and representing each node individually gives users rapid access to their details. Moreover, it provides a view into lower levels that is lost by flattening the hierarchy. The interaction connecting ResultMaps with other components of their environment guides users from text listings to ResultMaps and from ResultMaps to the listings (see Figure 1.1).

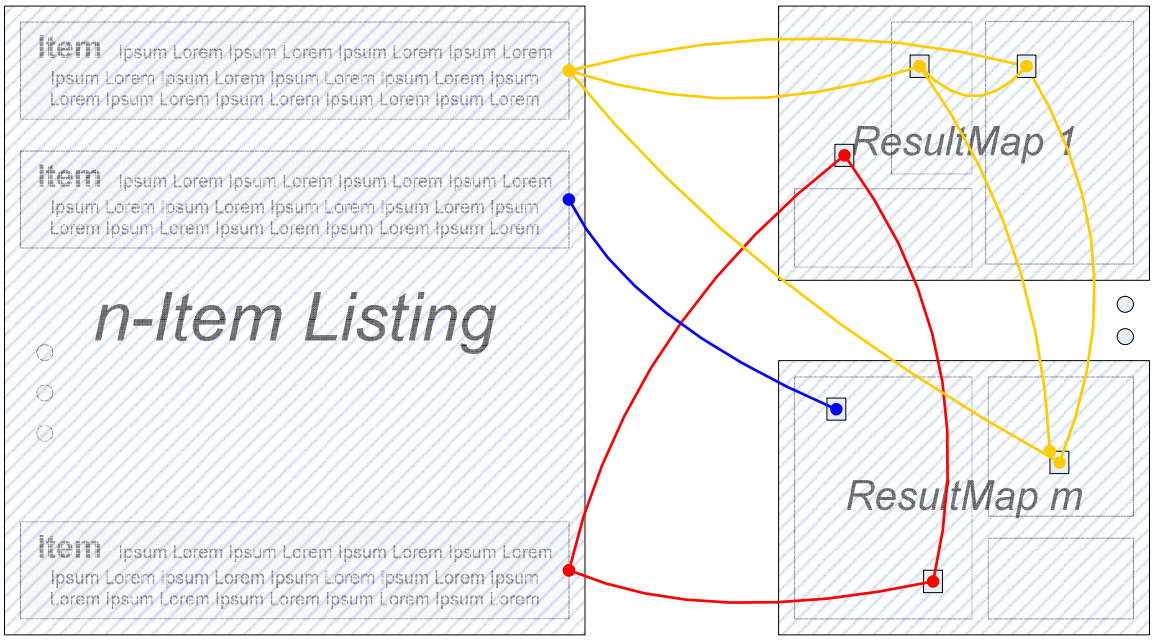

Figure 1.1. Conceptual diagram of ResultMaps. Lines indicate an interactive connection between an *n*-item result listing and *m* distinct ResultMaps. An item may or may not have an equivalent representation within any one ResultMap, and may have more than one in cases of multiple categorizations.

The latter transition is significant: the ordering of search results strongly biases user selections toward the top of the list regardless of the actual quality of the results: users essentially substitute the judgment of search engines for their own [38]. As a result,

users likely never even see highly relevant pages or documents that are ranked low on a SERP. This is especially problematic in the restricted context of a digital library, in which editorial control can entail few low-quality results, and link structure is less helpful in determining relevance. When no documents are specious, it is more likely that useful results will occur lower in SERP rankings. ResultMaps provide an alternative representation of results in which a more salient feature—like topical classification—can call attention to relevant results that are not highly-ranked in a SERP. This commentary applies just as well to faceted browsing systems, which often order or group items alphabetically. In addition, since the initial state of a faceted system is an unconstrained look at the entire result dataset, there is even more opportunity for interesting off-screen items.

ResultMaps also can facilitate user discernment of interesting data features, such as outliers and clusters, or relationships between attributes in faceted applications. While facets are typically described as 'orthogonal,' that applies to their semantic rather than their statistical relationship. As a result, significant and interesting correlations are often present between facets. Cluster and outlier detection are straightforward visual tasks using ResultMaps. Discovering correlations is fundamentally a more difficult task, but the ability to compare ResultMaps and their hierarchical representations assists users in discovering such relationships.

ResultMap consistency could also have beneficial ancillary effects: exposing a stable representation of the entire information space with every page view provides a means for users to gain additional knowledge about repository content as a side-effect of

searching for perhaps specific documents (e.g., breadth/distribution of topical coverage). That kind of knowledge can be useful for future information-seeking tasks.

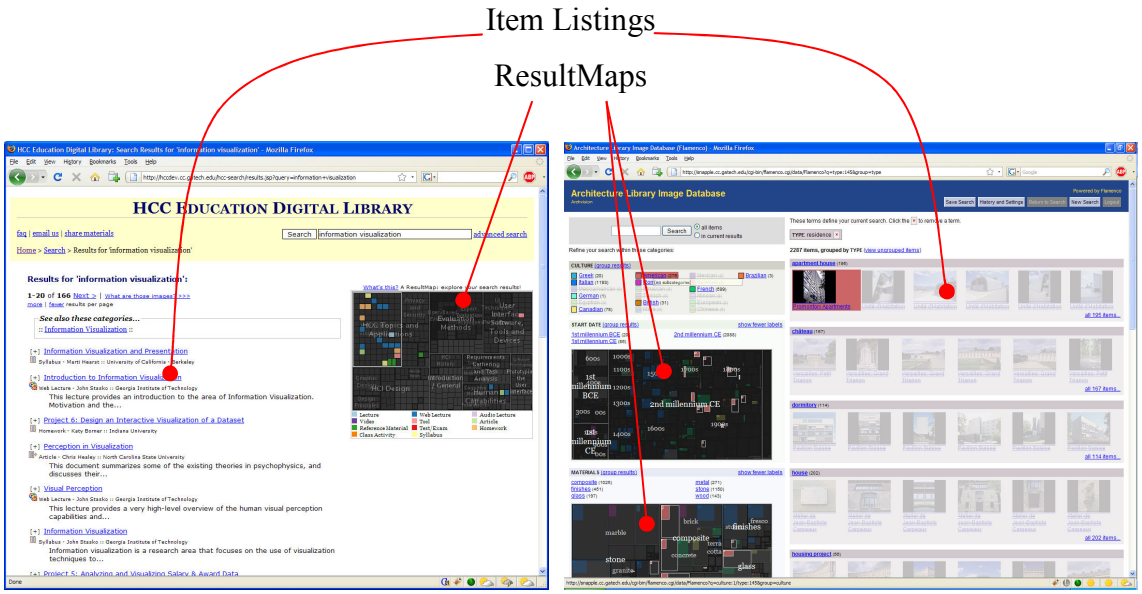

Figure 1.2. Instantiations of the ResultMap concept in search engine result page (left) and faceted navigation contexts (right).

We have applied ResultMaps in both SERP and faceted metadata contexts; Figure 1.2 shows screenshots of our instantiations of the ResultMap concept. We have conducted a series of experimental lab and field studies, which have provided some evidence supporting our claims above and also yielded a number of insights into the design and evaluation of infovis systems in this area. We have also developed formalisms of faceted browsing environments (using entity-relationship and relational models), which suggest further implications for the design of infovis systems for DL search environments.

1.1 Research Statement

We summarize our claims in the following thesis statement:

> ResultMaps (RMs) constitute a lightweight visualization mechanism for digital repository search systems. They provide a means for contextualizing repository content, providing several prospective benefits, while not impairing usage for uninterested users. Empirical studies of their usage, along with models of faceted environments, suggest a set of implications for the design and evaluation of future systems in this space.

1.2 Contributions

We support our thesis statement with three classes of contributions: empirical experimental results with the ResultMap-based systems we have built [26]; data and query formalisms for faceted systems [27]; and design guidelines derived from the above. We discuss each of these contributions in turn.

1.2.1 Experimental Results

We have conducted a series of evaluations on both our SERP and faceted ResultMap (RM) implementations: two formative, two formative/summative and two summative. In those experiments, we examine the following research questions:

1. How does adding SERP RMs affect user performance on DL search tasks?
2. How does adding RMs affect subjective impressions—such as satisfaction and engagement—of DL interfaces?
3. Do RMs yield a greater level of knowledge about the overall content in digital library as an ancillary effect of normal usage?
4. How do RMs affect query string characteristics over sequences of queries and other types of user behavior?

5. How do faceted ResultMaps affect subjective attitudes of faceted DL interfaces?
6. How do faceted ResultMaps affect the incidence of data insights (e.g., identifying data relationships such as correlations between facets)?

These questions are examined in 6 empirical studies, summarized in Table 1.1.

Table 1.1. Summary of empirical evaluations.

Name	Implementation	Purpose	Technique	Duration	Site	Research Questions
R1	SERP	Formative/ Summative	Experimental	1 hour	Lab	1,2,3
R2	SERP	Summative	Experimental	1 hour	Lab	1,2,3
R3	SERP	Summative	Log Analysis	6 months	Field	4
F1	Faceted	Formative	Think-aloud	1-2 hours	Lab	5,6
F2	Faceted	Formative/ Summative	Quasi-experimental	1 hour	Field	5
F3	Faceted	Summative	Quasi-experimental	6 weeks	Field	5,6

Our statistical results show that in some circumstances SERP ResultMaps yield better performance and user preference ratings; that faceted ResultMap ratings against a control condition are in a bimodal distribution, and that in all cases ResultMaps have no negative effects on preference or performance. These experiments and their results are discussed in the latter portions of Chapter 4 and Chapter 5.

1.2.2 Formal Models of Faceted Navigation Systems

We conducted a detailed review and characterization of extant faceted navigation systems, suggesting several relevant dimensions in which they vary. Based on that review, we have constructed entity-relation (ER) and relational models that describe the data and queries supported by faceted navigation systems. These models suggest new areas of exploration in the faceted design space by examining how the models might be made more general and how systems might be extended to support all aspects of our model. Furthermore, we can use these models to characterize both the supplementary

information that faceted infovis augmentations provide and the necessary costs (in terms of data queries). Our models are detailed in the first part of Chapter 5 along with their implications for faceted infovis; the review on which they are based is in Chapter 2 (section 2.4, p. 16).

1.2.3 Design Guidelines for Search Visualization

Our empirical and theoretical contributions form the basis for synthesizing design recommendations for online search visualization systems. We divide recommendations into those relating to the design of the system itself and those relating to the design of evaluations of such system. We summarize some of the more significant guidelines here:

<u>System Design</u>

- Visualizations that augment already complex base systems must be careful not to overload users—especially novice or non-visual users.
- Use progressive disclosure to hide complexity, but enable fast access to visualization enhancements.

<u>Research Design</u>

- Use within-subjects designs whenever possible, but develop and use more rigorous criteria for matching tasks between different conditions.
- User interest in datasets can be in the data items themselves (direct interest) or in the patterns the items form in the dataset as a whole (analytic interest): ensure the design of the search tool is targeting the same activity as the evaluation user and/or tasks.

We present evidence supporting these contributions in the following chapters. We begin by surveying some of the previous related work, which motivates our approach, before providing the details of our system design, experimentation and discussion.

CHAPTER 2:

RELATED WORK

Several related research areas influence this work: digital libraries and studies of their usage have shaped the creation of our own DL testbeds and how we study their usage. Library science, information retrieval (such as faceted navigation) and information visualization have clear implications for this work. Hierarchical structure is a pervasive organizational paradigm, cutting across all of these areas. This chapter summarizes relevant research from each of these areas as it relates to our effort and concludes by commenting on what differentiates this work from prior art.

2.1 Digital Repositories

In the span of a few decades (co-incident with the rise of the WWW), online digital libraries have moved from novel curiosities to key resources used as a matter of course in everyday activities. We surveyed a series of these repositories, classifying them by their purpose (educational, research or reference); audience education level; content location; content topical breadth; organizational schemes; and organizational granularity [24]. That survey identified a need for the HCC EDL (and later VADL) based on the relative lack of discipline-specific digital libraries for HCI educational materials. We also noted a lack of diversity in organizational and exploratory systems in use with live repositories.

Other research supports the idea of creating more specialized browsing constructs. Sumner and Dawe have examined the effects of educational digital library content

presentation on reuse [109], arguing that contextual information about library resources and the composition of those resources are important considerations for library design. Sumner and Marlino have also found that even teachers can have difficulty understanding how characteristics of particular resources are connected with broader aspects of the field [110]. In their evaluation of different types of hypermedia architectures [96], two of Salampasis and Diamantaras' conclusions are to 1) offer affordances for multiple information seeking strategies and 2) support parallel, interleaved use of those strategies. They also note that "the ease of use of the simple click-and-go-to interaction model introduced by the Web and the consistency of its interface appears to be more attractive for most information seekers." In earlier work, Xie came to similar conclusions in her study of library user behaviors [121], identifying support for opportunistic interaction and information-seeking strategy shifts as ways to improve information retrieval systems.

2.2 Information Visualization

Information visualization (*infovis*) is canonically described as "[t]he use of computer-supported, interactive, visual representations of abstract data to amplify cognition" [18]. Though comprising a huge number of systems and techniques, there are theoretical classifications that provide common ground within the field. Panning and zooming and focus+context are two such concepts. For example, panning and zooming interfaces render data at a single detail level at any one time, but data must be at a lower level of detail (LOD) to see it all at once; to see a high LOD near any one point other data must be cropped.

In contrast, Furnas' treatise on generalized fisheye views [34] discusses *degree of interest* (DOI) functions and how they can be used to render data of interest in detail

(focus) while retaining all of the other available nodes (context) at lower resolution. For hierarchical data, focus+context systems have the benefit of showing the overall structure of the data while allowing users to investigate specific areas of interest in more detail.

2.2.1 Hierarchical Information Visualization

Considering its pervasiveness as a knowledge and data structure, it is no surprise that information visualization is greatly concerned with different approaches to rendering hierarchical information. The traditional node-link tree structure is one of the simplest and most common representations for hierarchical data: it generally roots the tree at the top of the drawing space with nodes laid out successively below the root. The space requirements for the drawing area become large even with a relatively small number of nodes, however, leading researchers to present alternatives.

But despite its drawbacks, the traditional method is very familiar to most users. As such, many visualization systems leverage that familiarity by playing off the standard node-link look. ConeTrees [94] render trees in 3 dimensions and use interactive animation to assist navigation, improving screen usage efficiency. They also allow users to show or hide subtrees, but also introduce 3D occlusion as a potential concern. The SpaceTree exploration system [88] is a 2D system that achieves improvements in screen usage efficiency by the selective rendering of lower-level subtrees. It collapses large subtrees by default into proportionally smaller triangles, opening them only when user is interested in (i.e., interacts with) them.

Another alternative is to remove the constraint of top-down node layout. The Hyperbolic Browser does this by rendering a node of interest at the center of the display and radiating the rest of the tree outward [62]. It lays out the tree in a hyperbolic plane

and maps it onto a Euclidean unit circle, resulting in a tree that renders nodes of interest (near the center) in greater detail than those of lesser interest (at the edges of the circle)—a prototypical focus+context approach. User interaction can smoothly bring a node from the periphery to the center. Because any size tree is rendered within the same Euclidean space, this method is space-constrained with increasing numbers of nodes. Other space-constrained radial techniques use conventional geometry [8, 23, 106].

The treemap is a space-constrained, space-filling tree layout technique that represents tree data within an arbitrary rectangular region [52]. It recursively nests subtrees within the space contained by their parent with leaf node areas potentially corresponding to some data attribute. Because the rendering area is filled by the leaf nodes, treemaps are generally most useful when the leaf nodes themselves are of primary interest rather than the hierarchical structure as a whole. Treemap applications include the visualization and navigation of personal digital stores [104], business data tools [114] and stock markets [115].

2.3 Information Search Interfaces

Information seeking behavior is a complex human activity, and one that varies dramatically with system capabilities and users' model of those capabilities [73]. The theory of information foraging models that behavior on organisms seeking food [85], and suggests that users look for 'patches' of information, and move to new patches when the density of new information in given patch drops below a threshold. That kind of behavior can describe *exploratory search*, in which users satisfy general information goals through the combination of direct query and broader browsing strategies.

Information scent is a derivative of the general foraging theory that concerns how users perceive the value of a remote information source (e.g., via hyperlink text) [84]. In the context of exploratory interfaces, information foraging and scent theory suggest making clusters of related data clear and facilitating the process of finding new clusters of interest. The ScentTrails concept uses knowledge about the information scent within a data repository to highlight links on a page according to a user's keyword query [82].

Many search result visualization systems also work in concert with clustering algorithms, especially when the information space is extremely large or unstructured. The NIRVE system, for example, is a search result visualization system that displays documents and document clusters in an interactive 3D globe environment [101]. However, evaluation showed that a text search result listing outperformed both the 3D and a 2D version of the NIRVE system. LVis is another clustering visualization system with 2D and 3D versions [15]; it is also intended specifically for (image) digital libraries. Again, evaluation users performed less well with the 3D version. The 3D Cat-a-cone environment [43] allows users to search or browse medical document through multiple categorizations, showing their classifications in a 3D ConeTree representation.

The Pacific Northwest Lab's SPIRE system also uses clustering to extract common themes, and includes several visualization components [120]. Its Themescape component is an abstract 3D landscape depiction of a document space, with arrangements of hills and valleys representing the relatively strength of various themes in the document corpus and how those themes interrelate. The xFind system is a complete information space architecture, including a search client with both a scatterplot and a 'thematic clustering' display called VisIslands [7]. VisIslands is a 2D tool that borrows from

Themescapes, but visualizes documents matching specific search queries instead of the entire information space.

Kohonen (or self-organizing) maps [60] are visually similar to treemaps, but use neural networks to cluster documents into semantically similar regions. Lin applied Kohonen maps as a way of contextualizing digital library search results [66] in a manner that is similar to our search engine result page (SERP) application. Chen et al. evaluated a similar system, finding it less effective than either browsing or searching the Yahoo portal [19].

2.3.1 Directed Search Interfaces

Directed search is a core problem in both library science and information retrieval. As long as query string-based systems have existed, researchers have tried to create meaningful context beyond simple relevance-based document lists through visualization. An early example is the VIBE visualization system, which shows how different queries or parts of queries influence relevance [81]. Veerasamy and Belkin's visualization system addresses the same problem using a different visualization approach [112]. They compared their system to a text-based interface and found a few significant differences in favor of their visualization.

But as a WWW search paradigm has emerged and become more ingrained for even casual users [76], it is a challenge to assist WWW users without detriment to their expected usage environment. Government researchers discussed how to integrate web-based search and infovis relatively early in the history of the WWW [95]. However, they focus on applications in the then-fashionable (circa 1997) 3D VRML language rather than the page markup that has come to be standard. In contrast, Hearst's TileBars [41] do

well at minimally invading users' expected environment by augmenting the familiar text-oriented result list with informative glyphs. A modern commercial example is the Clusty search engine[4], which groups results into machine-derived categories alongside a traditional result list.

Kules' doctoral dissertation [61] largely focuses on features to use with automatic categorization of WWW search results, but does report a study of a treemap-based overview of search results, finding them comparable to outline-form overviews (and more effective than no overview).

2.3.2 Digital Library Interfaces

Using visualization for exploring digital repositories has been an active research area. A relatively early example is the Envision digital library, which includes a visualization system that places documents in a 2D grid according to user-selectable attributes [80]. The CitiViz [55] and EtanaViz [102] systems are Java-based applications for exploring digital libraries. Both use hyperbolic trees and scatter plots to visualize digital library contents. The hyperbolic tree uses either pre-formed or automatic clustering for its hierarchy; tree node sizes correspond to the number of relevant documents within that cluster or category. The ETANA browsing system also incorporates a faceted browsing interface. PARC researchers have used treemaps for access to personal digital stores [36]. They focus on issues with the transition from browsing the store to reading individual documents, with keyword query refinement cropping irrelevant documents from the visualization. Evaluation (an ethnographic account of use) is also left to future work. Klerx, Duval and Meire have implemented a

[4] http://clusty.com/

treemap-based visualization interface into the Ariadne repository [57]. However, there is no search facility within their tool and there is no application of their visualization to search results. Also, they did not report any form of evaluation.

2.4 Faceted Metadata and Navigation

Faceted classification (FC) is the process of categorizing a set of items into multiple independent (potentially hierarchical) taxonomies, and has been present in library science for over 50 years [92]. The FC process itself is outside the scope of this work: we focus on its product, *faceted metadata*. User interface (UI) tools into such data are called faceted browsers, and the type of usage they encourage is known as *faceted navigation*.

Employing FC results in a set of items into multiple independent taxonomies: for example, architectural works might be classified by their architect, location, construction materials, etc. These classifications are known as *facets*, and the collection of classification data is *faceted metadata*. The specific category labels within a facet are *facet values* (e.g., a material facet for a building might have values of stone, wood, metal, etc.). Facets values can be arranged hierarchically (e.g., stone → marble → Italian marble; stone → limestone). Each item may have multiple classifications within a facet or none at all. Some useful item metadata lies outside the faceted structure—for example, building name is unlikely to be incorporated into a faceted structure. We note this only to emphasize that that all item metadata need not be stored within a faceted structure.

Faceted classification is physically problematic, since an object can be stored in only a single location according to one classification—shelving books according to the

Dewey Decimal system, for example. But with computing and especially online databases and hypertext, using multiple logical taxonomies is possible. Thus faceted classification has seen tremendous growth in both research and commercial settings. These systems are commonly termed faceted browsing systems or faceted metadata UIs, or faceted UIs in short.

Faceted interfaces are often also related to the concept of interface *flow*, which is usually described in terms of the feeling of engagement and pleasure associated with executing a task [10]. The concept was studied in a computer mediated context by Ghani et al. [35], who found that perceived control, task challenge and requisite skill were factors linked to the flow experience. Faceted systems are often linked with flow because of the ease of switching between searching and browsing behaviors and the consequent control over the system [42].

It is perhaps indicative of some innate appeal that faceted navigation concepts have appeared in various guises elsewhere in the computing literature. The dynamic query concept generally [3] and specific applications like the HomeFinder specifically [118] have strong faceted components: their emphasis was on the benefits of real-time querying and selection on data exploration, but the underlying presence of multiple independent metadata features is a requisite feature. Likewise, Hieraxes [103] use different categorical or hierarchical attributes for each of its 2D scatterplot axes, making it in essence a faceted display of search results. The Attribute Explorer [111] bears many similarities to the HomeFinder system—including an essentially faceted view of data—and focuses even more on the information exploration via augmented graphical selection slider widgets.

The FacetZoom faceted navigation widget has some similarities to both to Hieraxes and to ResultMaps [28]. FacetZoom is also designed specifically for hierarchical facets. Hierarchy levels are rendered in a vertical stack, with the top row showing all prospective choices and lower ones indicating previous selections. If cell sizes are proportional to the number of items in each category, this method amounts to a stack of 1-dimensional treemaps. But FacetZoom purposely shows only a portion of the hierarchy rather than the entire context—its motivation is oriented toward navigation rather than visualization and analysis.

The Cat-a-cone interface mentioned previously [43] has faceted features: though in its published application it used a single hierarchy, documents could be classified under multiple categories. Microsoft Research's FacetMap system is a treemap-inspired exploratory search tool for faceted data stores [104]. Each facet is represented in a treemap, and user selections or query are interactively updated within all of the treemap views. Facet selections refine the information space by pruning documents; when space permits, individual document surrogates (i.e., thumbnails) render. The follow-on FacetLens system is a generalized faceted navigation framework with additional features [63], but until its official publication it is difficult to judge its complete capabilities.

Dimensional data modeling (see Chapter 7 in [30]), an active topic in the area of data warehousing, employs the star and snowflake schema patterns. In a star schema, rows in a central fact table are mapped to one or more dimensions of interest, which can then be used to select a fact or set of facts according to criteria from the dimensions. Snowflake schemas allow dimension tables to have their own dimensions (introducing a tree- or snowflake-like pattern with the fact table at the root or center), which reduces

data redundancy but increases query complexity compared with the star pattern. Both of these patterns essentially represent another rediscovery of faceted classification, and as such many data warehouse tools (e.g., Online Analytical Processing [OLAP]) have implicit faceted aspects to them. But there are many systems, especially recently, that are explicitly based on faceted concepts and navigation.

2.4.1 Faceted UI Design Themes

Allen [4] and Pollitt [90] represent two of the earliest instances of explicitly faceted (or "view-based," in Pollitt's terms) software. Researchers have also developed a wide variety of research faceted browsing systems, including Flamenco [122], the Relation Browser [72, 124], mSpace [100], Longwell [2], Humboldt [58], Bungee View [29], Elastic Lists [108], Freebase Parallax [47], the Nested Faceted Browser [46] and BrowseRDF [83]. There are certain commonalities found in these faceted browsing systems. Data elements in faceted browsers are partitioned into a central set of interest items and that set's facets. We call the items of interest the *focus* data or items. The facets (and their facet values) exist as a means of filtering the focus item set. Most systems divide their interfaces between a main focus area and a secondary facet area; selections generally occur in the facet area and affect overall UI.

The differences between these systems also capture the breadth of possible faceted UI designs. We divide system design variances into three major classes: visual design, interaction design and structural design. Visual design refers to primarily superficial differences, such as UI widget placement. Interaction design refers to behavioral features that marry the visual design to the underlying data, such as UI widget

types (e.g., buttons vs. checkboxes). Structural design refers to the data that are exposed in the first place, such as how a system deals with hierarchical data.

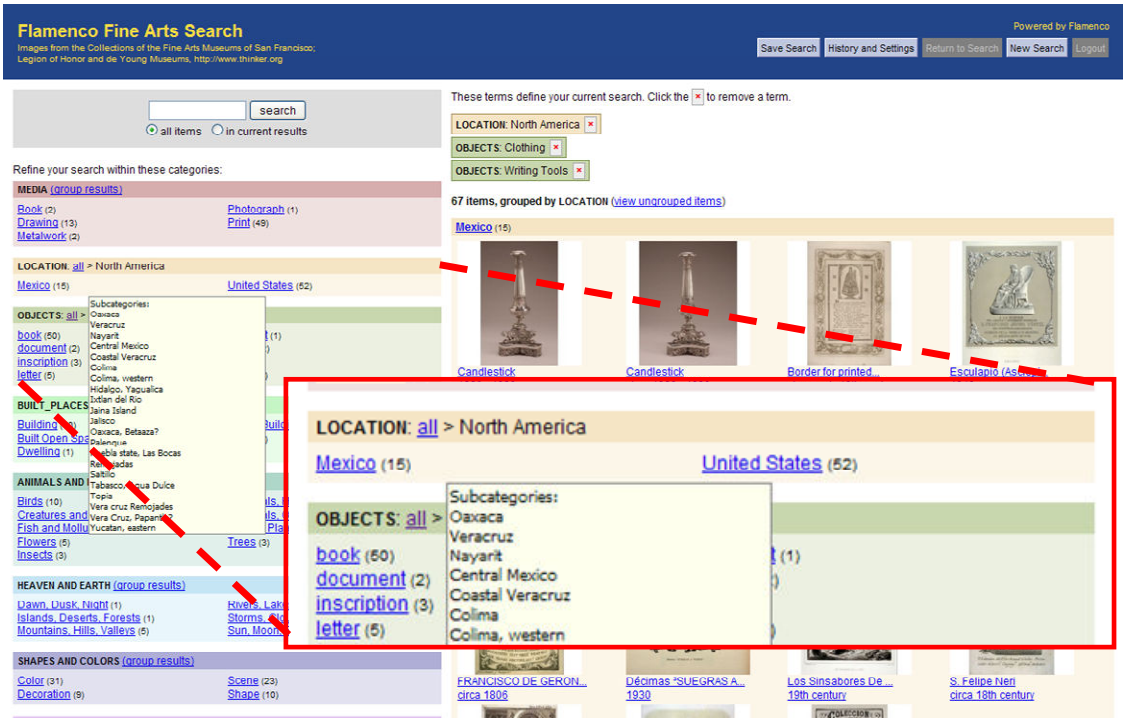

Figure 2.1. Flamenco browser on a set of fine arts images.

Visual Design

The core focus/facet relationship and its visual exposure is a basic feature of the data underlying any faceted UI. Using that relationship as a common denominator, we examine two elements of the visual design space: facet layout and cardinality data.

Vertical vs. Horizontal Facet Layout

An obvious UI design choice is how visually to relate the facet data to the focus data: that is, where to position the facet data within a window. There are two basic approaches: horizontal or vertical placement. This distinction is relevant because

vertical scrolling is much more prevalent, especially in web-based environments [77]. Thus, to conform to user expectations, faceted systems have limited horizontal and unlimited vertical space. Consequently, horizontal facet alignments limit the number of facets that can be displayed at any one time; vertical alignments have no such limitation (although requiring scrolling certainly introduces usability issues to consider). Flamenco (see Figure 2.1), Parallax, Humboldt, Bungee View and NFB all use vertical alignments. Elastic Lists (see Figure 2.2) and mSpace use horizontal alignments, and the Relation Browser uses both. The FacetLens system is a hybrid system, laying out facets according to a treemap-like layout algorithm; consequently its facets can appear horizontally or vertically (or both) depending on the specific facet values and distributions within a dataset.

Cardinality Data and Selection Preview

A preview of the results of a UI action is considered good UI practice. An important feature in information-seeking tasks is the size (cardinality) of the prospective result set. As such, most faceted systems present some indication of the focus item set size that would result from an action. This most often takes the form of a numeric count, which all systems but Humboldt and mSpace include. Earlier pseudo-faceted systems such as the Attribute Explorer [111] also emphasized this kind of cardinality data in its design. Interface elements like tooltips are also used to indicate the presence of any child categories (e.g., Flamenco in Figure 2.1).

Figure 2.2. Portion of an Elastic List of Nobel Laureates. Facet value brightness shows that selection's uniqueness with respect to the overall dataset. Facet value size represents its relative count within the results.

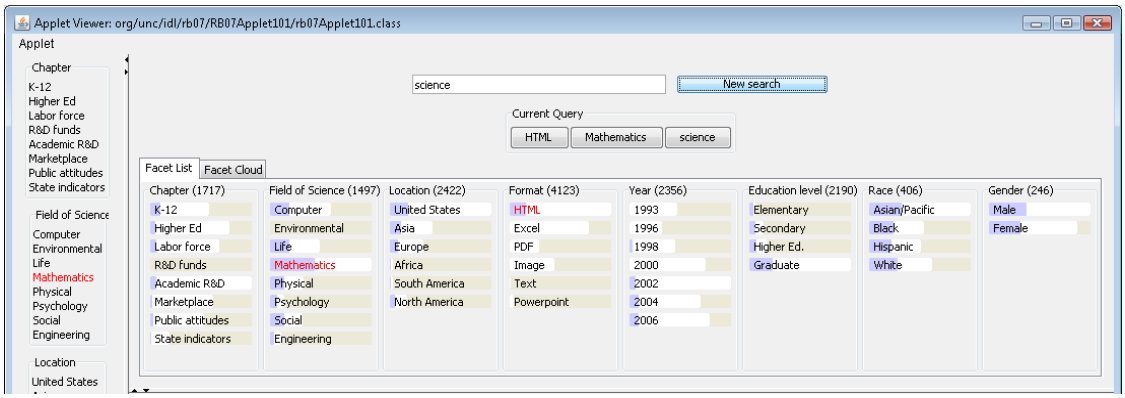

Figure 2.3. Relation Browser facets. White bar charts behind each facet value show their relative proportion in the overall dataset. Hovering over a value changes the blue portion of the bars in all values to preview the effects of that selection.

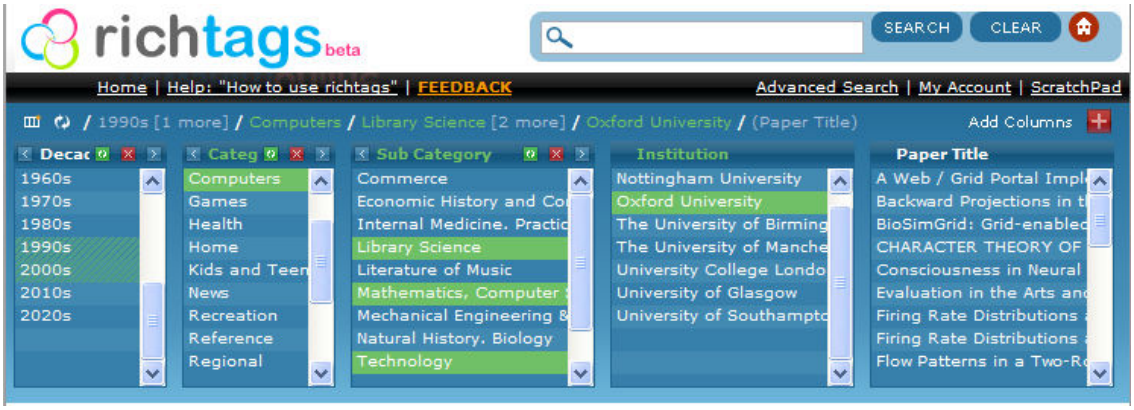

Figure 2.4. mSpace browser on a publication collection with *Decade*, *Category*, *Sub Category*, *Institution* and *Paper Title* facets. The user has made one *Category* selection, three *Sub Category* selections and one *Institution* selection.

But a raw count of items is not the only useful presentation of this data: a relative count—in relation to the number of items in the current result set or the total number of items in a dataset—can also be useful. The Relation Browser, Bungee View and Elastic Lists provide this additional data. In the Relation Browser each facet value link overlays a bar whose length is proportional to the number of matching items in the current result set, comprising a sort of scented widget [117]. Hovering over any facet value adjusts all facet value bars to show the counts if the prospective value were to be selected (see Figure 2.3). In Elastic Lists, the facet value widgets change size relative to their counts in the result set (see Figure 2.2). The brightness of a value also indicates a measure of the value's unusualness as compared to the overall dataset: brighter values indicate selections that have higher weight in the current focus set than in the overall data.

Bungee View combines these elements (see Figure 2.5): it uses stacked bars for each facet, with only a subset of the facet values present as labels (hovering over a bar section triggers a tooltip label). The bar lengths indicate global counts (i.e., total count of items classified in a category regardless of selections). Within each bar section, colored rectangles indicate (by height) the relative number of items matching that value within the current result set.

Interaction Design

Among the many possible variations in faceted UI interaction design, the behaviors in which we are most interested are those that 1) determine the queries a user can specify, and in turn 2) how the system changes to show the output of a query. We examine constraint selection (the former) and selection cascading (the latter) as two such design variations.

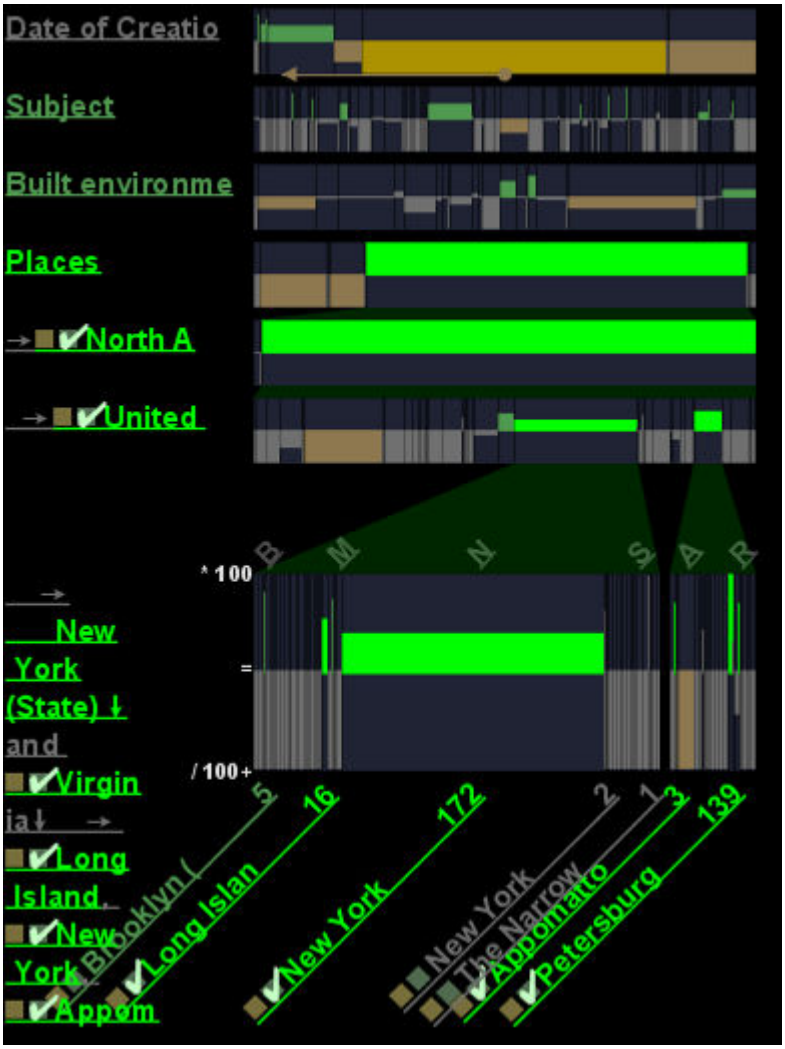

Figure 2.5. Bungee View facets for Library of Congress images. The 20th century has been negatively selected from *Date of Creation* and multiple selections have been made in the *Places* facet. Augmented stacked bar charts reflect global and relative item counts.

Selection Cardinality

Users can specify constraints to the system via any number of UI widgets: HTML links, buttons, checkboxes, etc. But the most salient distinction between these choices is whether the UI widget allows multiple selections. Only Flamenco generally limits users to single selections per facet, though it has some limited multi-query support: users can select multiple values from the same facet in the item details screen, but modifying those selections from the main browsing screen resets the selections to single values. In

addition, selecting multiple values is a conjunction of those criteria rather than a disjunction. The others implement multiple selections via multiple-selection lists (e.g., mSpace, see Figure 2.4), checkboxes (NFB; Bungee View; Humboldt), or other functionally equivalent operations. All of these methods act as disjunctive selections. Bungee View also allows negative selections (see Figure 2.5; the user has specified only non-20th century images): though logically redundant in this context, they are user-friendly by obviating potentially repetitive selections.

Selection Cascades

Wilson, Andre and schraefel characterize faceted browsers as directional or non-directional [21]. The key difference between these categories is that non-directional browsers treat facets independently of their order: selections in one facet affect other facets in the same way. Flamenco is a non-directional browser: selecting a value from the Location facet in Figure 2.1 modifies the content all facets with appropriate values (i.e., those that have a nonzero count). In contrast, in directional browsers the display order of the facets is significant: a selection in one facet only alters other facets in a single direction. This behavior is associated with horizontal layouts [119], and typically occurs left-to-right. It requires fewer queries since only facets to the right of the selection change. But facets to the selection's left become useless, since the system does not modify them with current values and so are redundant for subsequent selections.

mSpace is the only surveyed browser with directional behavior (it is also present in the popular iTunes interface). It also includes the backwards highlighting (BH) technique [119] that combines elements of the directional and non-directional approaches: a selection filters (i.e., removes non-matching) values in facets to the right

and highlights matching values to the left. Thus, BH modifies all facets as in the non-directional case, but differentiates facets based on their order as in the directional case. In Figure 2.4 the faint highlighting of 1990s and 2000s in the Decade facet are backward highlights from the selections to the right.

Structural Design

The basic focus/facet relationship is the common denominator between faceted systems, but there are many possible extensions to that core structure, which in turn afford various interaction and visual design differences. Here we discuss support for hierarchical and multi-focus datasets and, briefly, how systems expose those data features to users.

Facet Hierarchy

Some facet data clearly lend themselves to hierarchical structures—geographic location, for example. Others, such as time, can be modeled in a hierarchical fashion (e.g., subdividing time periods into progressively smaller units: millennium → century → decade). Any faceted system can indirectly support hierarchies simply by using separate facets for different hierarchy levels (cf. mSpace and the Category and Sub Category facets in Figure 2.4). But systems can also support hierarchy directly by allowing facet values to have different levels of granularity—that is, storing a parent/child relationship within a single facet.

Flamenco and Bungee View explicitly allow this kind of association. In Figure 2.1, selecting the North America value from the Locations facet has shown the next level of values in the Locations hierarchy. Selecting Mexico would show a further level of values in the hierarchy (listed in the tooltip). The system only shows the Mexico and

United States values (filtering out Canada, etc.) because only those values would result in matching items. In Bungee View in Figure 2.5, the user has selected the North America and United States values, showing new stacked bars underneath each of those parent values. Faint green shading connects a parent value to its child value sub-bars. The user has also disjointly selected the Virginia and New York values along with other sub-values. The disjoint selections create stacked sub-bars that are horizontally compressed to fit within the same area.

Indirect Facets

A tacit assumption we have made so far is that only one set of items has facets. Many frameworks make this assumption as well, but there is no reason that a set of categories cannot itself be described by other facets. We call such facets-of-facets indirect or distal facets. For example, an architecture dataset might contain information about architectural works (e.g., their composite materials; location; construction date; architect), and also have distal facets like cost and weight for materials, or nationality, gender, and alma mater for architects. Indeed, we could form indirect facet chains of arbitrary lengths (e.g., alma mater schools could have facets of their own, et al.). Furthermore, chains might form cycles in situations when items sets can be related in different ways. For example, the same set of people might be related to architectural works as architects, engineers, or benefactors.

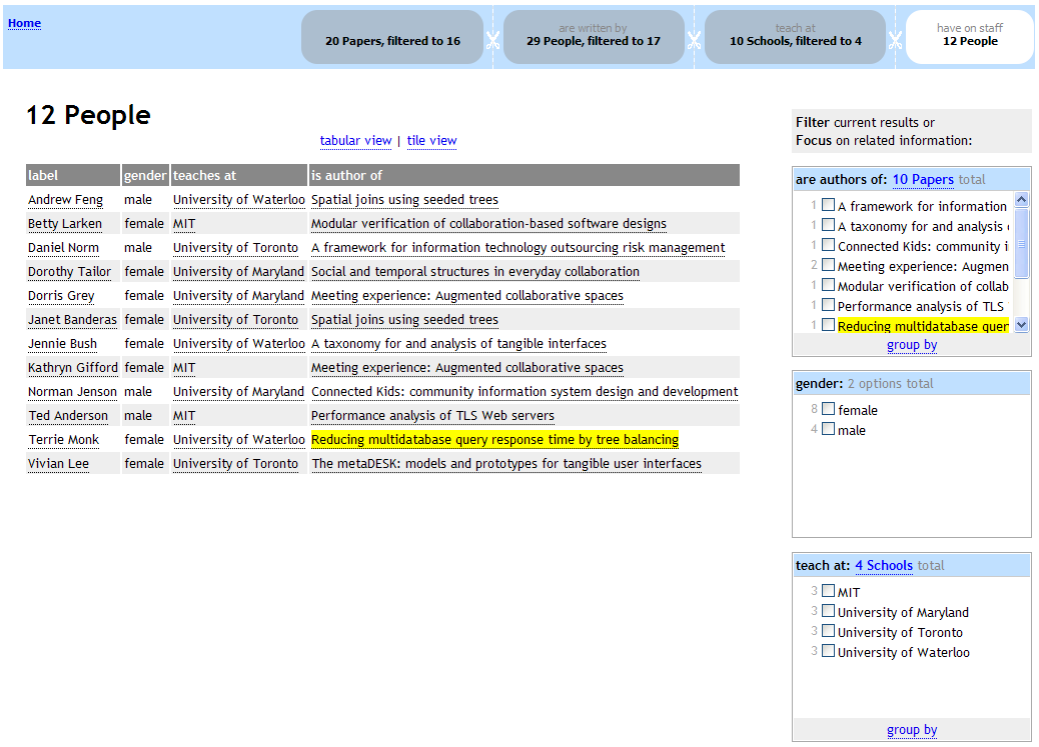

Figure 2.6. Nested Faceted Browser looking at a set of publications. The user has made a distal location selection of U.S. or Canada in the teach-at-school facet, filtering 10 schools to 4.

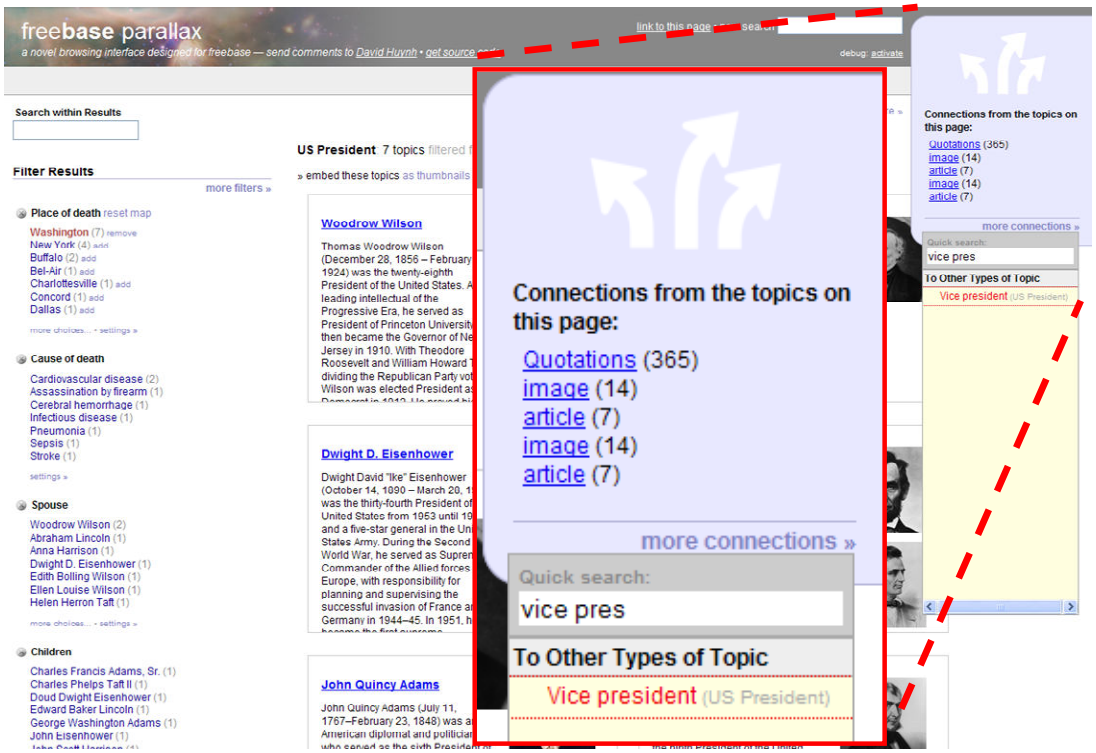

Figure 2.7. The Freebase Parallax browser, looking at U.S. Presidents who died in Washington, D.C. Other possible foci are at top right, such as Vice Presidents.

The NFB example is a good illustration of these issues. In Figure 2.6, the user has made several distal facet selections (figure top), including filtering the teach-at-school facet to those located in the U.S. or Canada. Using indirect selections is useful for complex questions: in Figure 2.6, 'who teaches at schools in specific regions?' More indirect possibilities include finding out if certain architects favor more expensive materials in their works: one could select high prices, which filter materials facet values first, then architectural works, before showing only corresponding architects. The Humboldt, Parallax, NFB all support indirect facets by virtue of being multi-focus browsers, the topic to which we now turn.

Single- vs. Multi-Focus

Many faceted systems also tacitly assume a single set of focus items: that is, a single group of items is the target of a user's information-seeking behavior. For example, most faceted retail systems have a single set of focus items: their products. But in a more network-like environment—such as the semantic web—users may have multiple potential target sets (Endeca calls this 'record relationship navigation' [6]). For example, in the architecture dataset we described above, users might be interested in focusing on either the architectural works, the architects themselves, or indeed switching their focus between the two. We call systems that support such actions multi-focus browsers. Others have termed the process of changing focus pivoting [58, 63] or refocusing [46]; we prefer the latter term.

The multi-focus nature of the Humboldt, Parallax (see Figure 2.7) and NFB systems directly stems from their semantic web and RDF research origins. The interconnectedness of that data makes multi-foci browsing almost a given, but also

impacts the way those systems approach indirect facets. Refocusing retains previous facet selections, causing selections from previous facets to become distal. In fact, in each case, this process is the only way to make distal selections. Because of this, distal selection can be somewhat difficult when the user is already looking at their item set of interest: one must refocus on a facet, make a selection on one of its facet, and then refocus back to the original items.

2.5 Evaluation

Evaluation of a software tool may take either a theoretical or empirical approach. In user-facing software, the empirical approach is often assumed because of the difficulties of usefully modeling the complex interactions between users and software. Moreover, the complexity and accessibility of those models that are robust stand as significant impediments to their use by non-experts. The GOMS (Goals, Operators, Methods and Selection rules) cognitive modeling approach [50] and information foraging theory [85] are two such theoretical approaches, both of which have been used on real-world systems [37, 84]. But even with tools to simplify theoretical model usage [51], empirical evaluation remains substantially more popular for information access tools.

Infovis task frameworks might be considered a theoretical evaluation system, such as Wehrend and Lewis' task classification [116] or Yi *et al.*'s interaction taxonomy [123]. But it is difficult to envision an evaluation along these frameworks being accepted as hard, summative evidence of a system's capabilities.

But even empirical evaluation of an information access tool is a difficult task, and is recognized as such within both information visualization [86] and library science [97]. Two components of any evaluation include choosing the unit of activity to analyze and

what aspects of that activity to measure. Both of these components are problematic in evaluating information-seeking software. One of the long-claimed benefits of infovis is that it enables users to "[ask] better questions" about data, but this is frustratingly difficult to prove [31]. Such hypothesized benefits typically occur at precise instances (e.g., so-called *aha* moments) within long periods of usage: but there is often a tradeoff in study procedures between detail and length (e.g., short-but-detailed lab studies vs. long-but-broad longitudinal diary or pager studies). Furthermore, precisely what to look for is also not trivial: how do concepts like "asking better questions" or "insight" translate into measureable events?

One way of organizing a discussion of the approaches to these problems is around the distinction between quantitative and qualitative data gathering and analysis. Note that these dimensions are independent: one might gather qualitative data but analyze it quantitatively, for example. We discuss each of the four combinations in turn along with representative examples of their use in infovis and information search evaluations.

Quantitative Data Gathering and Analysis

Quantitative experimentation and analysis, typified by the controlled lab study scenario, is the classic approach to scientific research. Precision and recall metrics have long been standard quantitative measures of the effectiveness of an IR system or algorithm. Perhaps unsurprisingly, quantitative assessments and metrics have also been prominent among both purely infovis [59, 62, 88, 101, 106] and digital library work [15, 17, 19, 82, 96, 112, 122, 124] as well. But there is a trend among this type of work of finding few (e.g., [17, 62, 96, 112]) or negatively (e.g., [15, 19, 101]) significant differences between conditions, especially on task performance measures. To some

extent, this is not surprising: if infovis purports to improve users' information goals, there is no reason to expect that users will necessarily be more efficient in reaching goals—especially if they change.

Controlled lab studies are not the only opportunity for quantitative data gathering and analysis. Software log procedures—most often recording user actions via instrumented software—also provide opportunities for statistical analysis of their results. However, because their often longitudinal, real-world usage typically cannot be as well-controlled as in lab studies, the output has greater potential for confounding factors. The problem of non-significant results is still an issue for even for long-term behavior logs (e.g., [54]).

Qualitative Data Gathering and Quantitative Analysis

The mixed results common to many lab studies have led researchers to explore more qualitative ways of gathering data that is still amenable to quantitative analysis. Self-reported surveys of subjective satisfaction are common (and more often result in statistically significant results). A significant issue is the use of *ad hoc* satisfaction surveys rather than validated instruments such as the QUIS [21] from the University of Maryland or the CSUQ [65] from IBM. The NASA TLX cognitive load instrument has also been widely deployed in HCI projects as a whole [40]. Beyond generic usability instruments, we have already mentioned the concept of interface flow. Measuring how well a system achieves this is a challenge; Ghani *et al.*'s approach [35] to inquire about user engagement and enjoyment of their usage period has proven promising and has been successfully adapted and deployed by other research teams [17].

A recent infovis trend has been toward what has come to be known as *insight-based evaluation* [87, 98, 99]. Drawing from the principle that infovis is about "insight, not pictures" [18], this method attempts to statistically analyze qualitative data about the qualitative insights that users gain into a dataset. But how does one measure the concept of *insight*? North provides some defining characteristics: insight is complex, iterative, qualitative, unexpected and contextual [79]. He suggests that experimental tasks should either be correspondingly complex or non-existent (i.e., user-generated rather than set benchmark tasks) to have hopes of facilitating such insight. In either case, insight analysis relies on subject-matter experts to generate a coding scheme to identify insights and rate their insightfulness from the users' findings. Rating insights according to such a scheme generates data that can be quantitatively analyzed, along with other insight-related factors: time to first insight, types of insights, etc.

Quantitative Data Gathering and Qualitative Analysis

A qualitative look at quantitative data has also proven useful in IR and infovis work. In most cases, this type of evaluation uses quantitative data with supportive qualitative data, such as behavioral statistics along with user quotes, to support a high-level analysis of a system. Qualitative analyses of uncontrolled survey or questionnaire deployments are also common. Salampasis and Diamantaras used a simple form of this evaluation: basic uncontrolled Likert scale survey data incorporated into a qualitative account of their system's ease-of-use and underlying data model.

Social infovis systems seem to employ these types of evaluations in particular, perhaps because the nature of their usage often makes it difficult to gather data in any kind of controlled fashion. The *sense.us* system authors, for example, used counts of

visual annotations, navigation mechanisms and coded user comments to buttress their claims about the social nature of infovis tool usage [44]. IBM's Many Eyes reused the same coding scheme to similar ends [113].

Qualitative Data Gathering and Analysis

Qualitative data and analysis have been common in the social sciences and have more recently become more common in the infovis, IR and digital library literature. In particular, casual or ambient infovis systems, like the Tableau machine [91] and the InfoCanvas [107], have used qualitative evaluations, which suit their use cases: non-analytic, long term, novice personal reflection. The developers of the Perseus Digital Library evaluated its use in a variety of ways, including participant observation, interviews and document analysis [71]. Though not strictly evaluative or about software, Johnston used a qualitative ethnographic procedure to examine—among other things—the information-seeking practices of intelligence analysts [53].

We have discussed evaluation within these four quadrants to emphasize the diversity of approaches and to differentiate data gathering methods from their analysis. However, we do not suggest that the various combinations are in isolation from each other; on the contrary, different data gathering and analysis methods complement each other by confirming or contradicting other results. For example, lab experiments commonly triangulate the results of quantitative data and inferential statistics using qualitative debriefing interviews (e.g., [82]), coded video of the experimental session (e.g., [59]) or surveys (e.g., [62]).

2.6 Discussion

The purpose of this chapter is to array our research decisions within the larger context of various fields. We use that context to distinguish our work from previous in some instances, and to justify our decisions in light of earlier results in others. There are similar uses of treemap [15, 57, 61, 80] or map-like [19, 66] approaches to search visualization. However, many works present only limited evaluation of their effects. Even simpler data graphical representations like bar charts [17] have only limited studies associated with them. In this work, we address the effectiveness of those visualizations directly as well as assessing ancillary effects on users such as overall repository knowledge.

We view the implementation context of our systems as important as well. Among exploratory search work, systems tend to be either web-based [13, 46, 47, 82, 100, 122] without any visualization component or include one, but rely on a separate application or plugin environments [7, 17, 29, 55, 101, 102, 104, 120]. There are exceptions [95], but none that target 2D map-like contextualization with our research approach. Using only standard web technologies has significant practical benefits: it serves as a proof-of-concept that interactive visualization systems can be built without relying on propriety technologies and integrates well within the WWW ecosystem (no plugin loading latency, bookmark-able state, standard hyperlink navigation, etc.)

We also find little discussion of how exploratory systems can leverage visualization to increase users' state awareness through preview and history. Plaisant et al. found providing query previews to have both subjective and objective benefits [89], and some of the faceted systems do provide some preview facilities (e.g., tooltips [122]

or interactive changes to data glyphs [17]). But that work has not been exhaustive, and there is little evidence indicating the utility of providing feedback on past choices rather than future ones. It is plausible that providing mechanisms for diagnosing the effects of the most recent facet value selection would be especially valuable in faceted environments.

Our brief survey of the various evaluation approaches influences both our overall method and the metrics we employ. We use a variety of data gathering and analysis methods, including: purely quantitative lab experiments; field data log analysis; observational think-aloud sessions; and field deployed quasi-experimental deployments of our systems. This diversity allows us both to triangulate our results and to increase the likelihood of finding evaluation method that are particularly well-suited to our usage contexts. Like other infovis evaluators, we focus less on performance measures and more on subjective measures (with validated instruments) such as engagement [35] and satisfaction [65]. Our later evaluations also attempt to solicit insight-related data using more open-ended task formulations.

Finally, though our survey of faceted systems is relatively small, we are aware of no systems with capabilities dramatically different from the combined feature set of our sample. The results of that survey in particular motivate our formulation of coherent formalized data and query models for faceted navigation systems. Though such models are useful to describe existing work, drive future extensions, and assist in designing and building faceted systems that improve on the state-of-the-art by combining features only available within individual existing systems.

CHAPTER 3:

DEFINING RESULTMAPS

We use the term *ResultMap* to denote a treemap with particular characteristics, which we now make explicit. Specifically, a ResultMap (RM) is a treemap that:

1. encodes a repository's full contents according to a hierarchical metadata attribute[5],
2. accentuates certain nodes as indicated by a query engine, and
3. interactively links accentuated nodes to corresponding entries in a text listing of query responses.

We refer to the metadata attribute used in a ResultMap as the *mapped attribute* or *mapped variable*. For example, a digital library whose contents have been categorized into a hierarchical topic taxonomy might use ResultMaps with a standard search facility. A ResultMap could use each document's position within the taxonomy as the mapped variable and highlight particular nodes according to results returned by the engine. Such a case is our initial deployment of ResultMaps, detailed in Chapter 4. Potential real-world applications might include Usenet newsgroup or eBay item search facilities.

Encoding the entire repository (rather than, for example, encoding only items that are relevant responses to a query) provides a stable context in which to accentuate query responses. The pairing of the data visualization and a traditional text display combines a familiar paradigm for exploring query results with a visual interface to the same data; the interaction between the two reinforces the pairing and allows the user to move between textual and visual processing of the same data.

[5] Using a flat categorization attribute is possible, but a questionable choice given that a stacked bar chart would convey the same information.

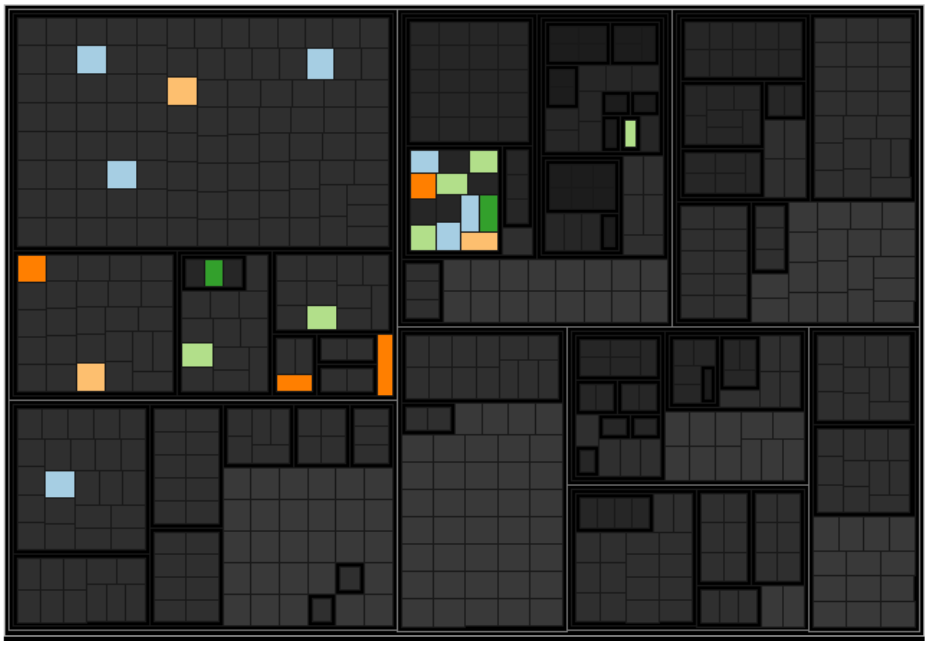

Figure 3.1. A treemap representation of (arbitrary) data that has a skewed distribution at lower tree levels.

Treemaps are especially suited for these requirements over many alternatives. In particular, treemaps are space-constrained and -efficient methods of representing complex hierarchies. Since ResultMaps are paired with text listings, economical use of display space is desirable. Constrained space usage means ResultMap screen space requirements are invariant with a given repository's size. Other representations can provide some of the same functionality for hierarchy; however, they typically aren't space-constrained (e.g., typical node-link tree diagrams), or space-efficient (e.g., hyperbolic trees [62]). Simpler approaches like data graphics or numeric indicators can give users a high-level indication of how a corpus is distributed across metadata values, but when lower levels of the tree are unbalanced that skew is obscured by simple summary displays. A treemap representation does not mask such skew (see Figure 3.1).

For example, consider a database of Nobel Prize winners that has a hierarchical metadata field for the continent/country of origin. A flat view of the data might show at

its top level that Europe has significantly more winners than North America. But a treemap of the same data—which renders the entire hierarchy—also shows that the winners are not distributed equally across the two continents: the United States claims over 95% of the North American winners, while the European winners are more evenly dispersed. Figure 3.1 does not show a real dataset (or the Nobel data in particular), but the top left cluster is analogous to the European winners—relatively equal distribution across a top level in a hierarchy—and the top middle is like the North American winners—one lower-level category (the United States) containing most of the items within the overall top-level items.

A potential issue with our solution is multiple categorizations: the treemap layout deal with strict tree data structures, which assume a node has at most one parent. However, in many hierarchical organizations a leaf node (or less commonly internal nodes) might have multiple parents—a single message that has been cross-posted to multiple newsgroups, for example. We deal with this case by assuming that semantically equivalent nodes are logically duplicated within the ResultMap input data structure, but retain their equivalency with respect to any interactive effects. That is, whenever an event is triggered on the node in one part of the hierarchy, the same event occurs in any other semantically equivalent nodes: for example, if a cross-posted newsgroup message is highlighted in one of its groups, it is highlighted in all of its groups.

3.1 Design Choices

We are faced with a variety of choices in our actual implementation of the ResultMap concept. Among them are the specific design choices of what attributes map

to what visual features, visual feature attributes (e.g., colors and shading), individual vs. collective representation and the specific treemap layout algorithm.

3.1.1 Layout Algorithm

The original treemap layout algorithm is known as *slice-and-dice* [52] because it alternates between cutting rectangular spaces horizontally and vertically. Two important properties of a treemap layout algorithm are the aspect ratio of the rectangles it generates and the stability of its layouts as the source hierarchy changes. Low-ratio rectangles (more square) are preferred since area comparisons are easier between nodes; stable layouts, in which relatively small changes to the hierarchy do not result in relatively large node placement shifts, are also preferred. Low-aspect ratio nodes are also preferable because Fitts' Law predicts they are easier to select than skinnier items [69]. Though conscious decision making likely dominates total selection time, once the user has decided to navigate to a given node, Fitts' Law should apply—hence, square nodes are preferable to thinner ones.

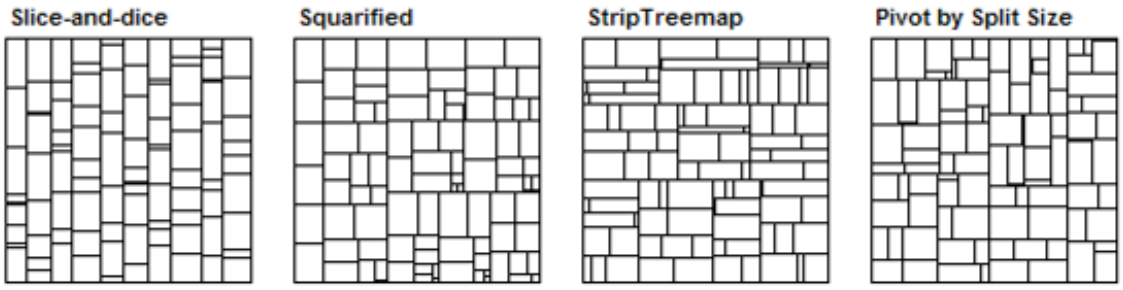

Figure 3.2. Comparison of slice-and-dice, squarified, strip and pivot treemap layout algorithms for same 100-node (10 nodes with 10 children each) tree. Screenshot retrieved May 2009 from http://www.cs.umd.edu/hcil/treemap-history/java_algorithms/LayoutApplet.html.

The original slice-and-dice method does not fare well on any metric (other than computational efficiency), and as a result has been superseded by a host of newer layout techniques, notably the squarified [16], pivot, strip [11], and Voronoi [9] methods (see Figure 3.2 for a comparison of the former 3 layouts with the original). Though the authors provide no quantitative data, their description of the Voronoi algorithm indicates that it is considerably more complex than the conventional rectangular ones. Integrating non-rectangular features with inherently grid-based web markup also poses a technical challenge. With these factors in mind, we use the squarified or strip layouts for our ResultMaps: squarified for metadata which has no inherent order and strip for that which does. Both algorithms generate relatively low-aspect ratio nodes and have acceptable execution times.

Changes occur to our data sources (digital library collections) on the order of a day rather than minutes or seconds. While we have noted that ResultMaps should have a stable context for query highlights, that chiefly refers to always showing the overall DL collection, not the specific collection items or visual arrangement *per se*. Constancy is preferable, of course, but it is more important to have it intra- rather than inter-session. In addition, studies have indicated (as does our own usage data) that the majority users over a period of time are first-time visitors [39, 71]—and as such, the inter-session stability of the ResultMap layout is immaterial most of the time.

3.1.2 Node Features

What individual nodes represent—i.e., a document or a category—is a basic design choice. We use hierarchical metadata for the requisite tree structure but leave leaf

node size constant[6] in our ResultMap implementation. But when leaf nodes represent individual documents (*individual ResultMaps*), intermediate category nodes have sizes corresponding to the number of documents classified within that category.

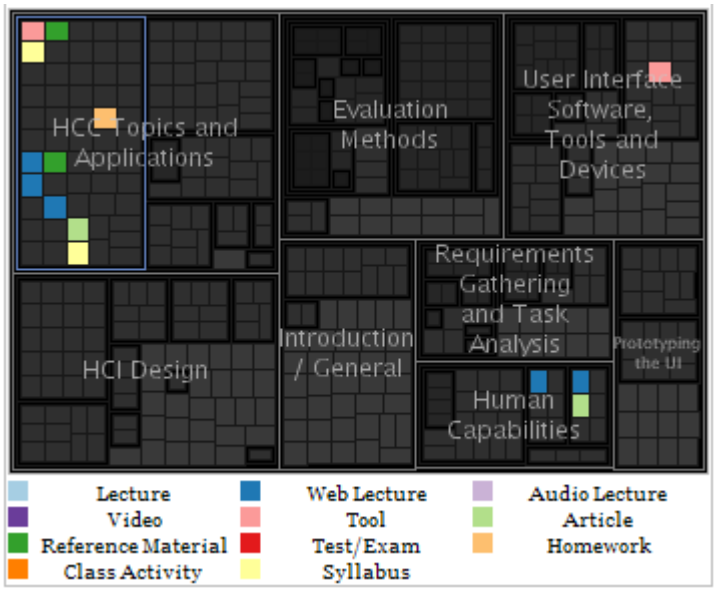

Figure 3.3. A ResultMap (with key) using the squarified layout algorithm, highlights via color saturation and un-weighted node size.

Moreover, we have relatively few quantitative variables—which would be suited to a continuous ratio attribute[7]—in our datasets that would be useful to encode as leaf node size. Search relevancy scores are one possibility, but would result in ResultMaps with unpredictable layouts and run counter to our goal of providing uniform context for query result highlighting. A suitable data attribute would be ratio variable and invariant—or nearly so—to any specific query. Data that is variant with query (e.g.,

[6] We use the term *node size* somewhat loosely to denote the degree-of-interest (DOI) multiplier. Strictly speaking, the DOI multiplier is constant, but actual node size can vary due to rounding and the category node borders that take space from the document nodes nested therein.

[7] Continuous interval variables are invariant to linear transformations; continuous ratio variables are invariant to any transformation. Only ratio variables are suitable to encode in node size since interval measures have no true zero point.

relevancy) introduces instability to the node weights and accordingly to any treemap layout. An attribute like price in eBay auction data is a likely candidate, but our data attributes do not have such an analogue. Popularity (as measured by number of downloads) is one option; however, we explicitly do not want to emphasize more popular documents in our DL to avoid having them drown out lesser-used content.

We color highlighted nodes and use grayscale for non-highlighted entries (see Figure 3.3). Brightness for both color and grayscale decrease slightly with increasing tree depth, which makes hierarchical structure easier to interpret. Node color (for highlighted nodes) may be chosen according to some metadata attribute from the dataset, which we term the *key attribute*.

The size limit of easily-distinguished palettes is generally considered to be at most a dozen distinct colors. Categorical or qualitative variables might easily have more than 12 possible values: in those situations, the best recourse is to consider whether key attribute values could be aggregated. For example, a *Country of Origin* attribute could be aggregated into continents or regions. Barring that, the designer might reconsider the utility of using that variable in the ResultMap if it forces the color representation scheme into something beyond useful perceptual bounds. Encoding ratio variables is less troublesome since they can be aggregated with arbitrary value bounds.

In any case, in a faceted setting (in which there may be multiple ResultMaps), it is also important that node color and size encode the same attributes across all faceted ResultMaps, so that the mapped attribute (i.e., the hierarchical attribute that is encoded within a specific ResultMap) is the only operative variable when comparing faceted ResultMaps. Since facets generally consist of the most useful metadata, the key attribute

for a ResultMap becomes the *key facet*, such that node colors across all faceted ResultMaps are determined by the values of the key facet.

3.1.3 Scalability Analysis

Since a stated goal of ResultMaps is to represent the whole of a digital repository, we would be negligent not to explore and quantify the representational limits of ResultMaps, especially ones that encode documents at the individual level. Fekete and Plaisant have addressed parts of this issue in demonstrating their million-item treemap [32], but only some of their work directly addresses our specific limitations. A discussion beginning with the essentials is worthwhile.

The naïve upper bound on the display limits of a treemap x pixels wide by y pixels tall is simply the total number of pixels $x \cdot y$. But element borders are often necessary (and are included in our implementation) to make the hierarchy clearer to the user—this reduces the available number of pixels. The exact number is dependent on the number of leaf nodes, internal nodes, and how the leaf nodes are distributed across the internal nodes. As a result, a mathematical model of frame-expended space—even one for a simple lower bound—is difficult to derive.

In our search engine result page (SERP) implementation detailed in Chapter 4, the ResultMaps are 350x233 pixels with a 2 pixel frame; our library corpus as of March 2009 has 90 categories and 585 documents (including ones with multiple categorizations). Under those conditions, frames (including the 1-pixel spaces between leaf nodes) consume 26.9% of the 81,550 total pixels, with an average node size of 10x10 pixels. We have found few explicit recommendations for minimum control size in the literature. The Apple Human Interface Guidelines [1] have no controls with dimensions smaller

than 7 pixels. An informal survey of the default Windows XP window manager found nothing smaller than 6 pixels. We use the smaller of these since ours is an upper bound discussion.

Thus, using a minimum mean desirable node size of 6x6 pixels, individual ResultMaps should scale without modification to about 1700 documents. Assuming a minimum 3x3 mean size, they can accommodate just fewer than 6800 items. Fekete and Plaisant suggest eliminating frames altogether in favor of slightly-shaded rectangles, which may be sensible when a ResultMap's mean node size drops below a certain threshold.

Individual vs. Collective Representation

Our discussion has so far made the assumption that documents are represented individually level and are equally weighted. We can boost ResultMaps' representational power by relaxing those assumptions, which is warranted for larger repository sizes (or smaller display areas). But we should frame our discussion around our requirement to use meaningful and consistent context for highlighted nodes.

One option is continuous, dynamic DOI emphasis—most often represented by a fisheye lens distortion. Magnifying the nodes around the cursor means the effective size of all nodes is increased while the actual node size could be as small as a single pixel. Under those conditions our SERP ResultMaps could expand to over 61,000 items. However, users can find the distortion inherent in fisheye views disorienting; furthermore, using standard web technologies (i.e., client-side JavaScript or AJAX) to implement such a feature is problematic from a pragmatic perspective given the JavaScript performance capabilities of 2009-era web browsers.

Static DOI emphasis of highlighted items is another alternative: increase the DOI multiplier of the highlighted nodes within the treemap layout algorithm, which increases their size relative to the other (contextual) nodes. This has the advantage of being consistent with our goal of representing the entire document space. However, different combinations of node highlights produce dramatic variation in layouts between queries, especially when using a relatively unstable layout like the squarified algorithm. We could reduce instability by offsetting DOI increases with DOI decreases in the nearest neighbor nodes, as in [56]. In that case, the resultant node size increases for highlighted items are made at the expense of nearby nodes, so dramatic layout shifts should be minimized. This system fails in densely-highlighted circumstances such as faceted ResultMaps, though users' interaction is merely not improved rather than the representation itself breaking down.

We could also address the effective rather than the actual size of a highlighted node. A conceptually straightforward procedure is to partition the ResultMap interaction space into a Voronoi diagram with each highlight as a focal point, and use the Voronoi partitions as the activation area for each node. We can also trade efficacy for simplicity by using best-fit bounding boxes rather than the exact partitions. In either case the effect is making node interaction less difficult, but with the advantage of a consistent visual representation regardless of which nodes are highlighted.

All this being said, our preferred option is to not require individual representation of each document in a dataset, but instead to collectively represent documents by bucketing leaf nodes together so that one node area represents multiple logical documents. Specifically, we bucket leaf nodes within their parent categories by their key

attribute/facet values. This is equivalent to rendering an individual ResultMap that sorts leaf nodes by key values and halting the layout one level above the leaf nodes.

Somewhat like the Information Mural approach [49], this has the obvious benefit of multiplying the number of nodes we can represent. However, a collision occurs when we need to highlight multiple nodes in the same bucket. We address this by highlighting all relevant documents in a list from their ResultMap representation. While this precludes acting on a ResultMap surrogate identically with a document's list representation (e.g., clicking on it to show details about a document), this tradeoff is warranted to support real-world library sizes.

The scalability of this approach is thus bounded by the number of categories in a classification rather the number of documents: corpora of any size are supported for reasonably-sized classifications. This method also has the advantage of allowing a system to transition gracefully from individual to collective representation: the overall positioning of the category nodes does not change. We call ResultMaps using the former strategy *individually represented* or *individual* ResultMaps and the latter *collectively represented* or *collective* ResultMaps.

The decision about at what depth to stop rendering the treemap has several possible answers: in the version above, rendering stops at the bottom-most category nodes (the depth of which vary unless the categorization hierarchy is perfectly balanced). In our last implementation, we use an different approach by picking a static depth (e.g., 2 or 3) and only render category nodes to that depth. This method simplifies ResultMap visual complexity (by reducing the number of rendered nodes, which are thus larger within the available space) at the cost of data about how items are distributed across the

lowest-level categories in a hierarchy. A potentially useful approach (which we have not implemented or tested) would be to mix the latter two methods: render category nodes at any depth as long as the resulting nodes are above some size threshold. This dynamic rendering would also sacrifice some distribution information for visual simplicity, but do so in sections of the ResultMap where it would be most effective (i.e., very deep or less-populated parts of a categorization hierarchy).

3.2 Task and Interaction Classification

There are a variety of taxonomies of both task and interactions in the infovis literature. We choose two of them as a basis for classifying the tasks and interaction techniques which ResultMaps support. Doing so gives us an explicit and more formalized notion of the kinds of use environments at which we have targeted ResultMaps. We choose these two taxonomies in particular because they have the advantage of building on previous attempts, and are abstracted from particular applications.

Yi et al. have categorized interaction techniques commonly used in infovis systems [123]. Though they discuss applications that are generally more complex than what we have described, their classification is still applicable to our ResultMap implementation. Their interaction classes are *select, explore, reconfigure, encode, abstract/elaborate, filter* and *connect*.

We expect ResultMap use to consist either of 1) identification of a result from a result listing with follow-up use of the ResultMap for more information or 2) identification of a result in the ResultMap with follow-up use of the result listing for more information. The former case can be described as a *connect* operation ("show me

related items") because the transition reveals information about how a node characteristic is related to the overall repository and to other nodes in the current result set. The latter transition is an *elaboration* ("show me more detail") because it produces detailed text about a document as well as the underlying data resource. Though they are not infovis *per se*, parts of our implementations that move to different dataset results (e.g., links to other search results) might also be described as *explore* ("show me something else").

Amar et al. [5] is one of several classifications available (see also [116]) for describing users' low-level purposes (rather than the actions used to achieve them) derived from an affinity diagram analysis. That analysis yielded ten tasks: *retrieve value, filter, compute derived value, find extremum, sort, determine range, characterize distribution, find anomalies, cluster* and *correlate*. Like Yi et al., this work proceeded from a goal of abstracting the classification from specific visual representations. As a result, we use this task taxonomy as a suitable companion to the interaction classification above.

We have already mentioned our intention for ResultMaps to be useful in identifying clusters, outliers and overall relationships between dataset attributes. All of these tasks have obvious corollaries in the classification:

- Cluster ("given a set of data cases, find clusters of similar attribute values")
- Correlate ("given a set of data cases and two attributes, determine useful relationships between the values of those attributes").
- Find Anomalies ("identify any anomalies within a give set of data cases … e.g., statistical outliers")

All of these tasks also have some element of *Characterize Distribution* ("…characterize the distribution of [an] attribute's values over [the set of data cases]") to them, although the authors' full description explicitly cites quantitative attributes only.

Correlate and distribution tasks can be particularly complex, so it is useful to review a few infovis approaches that also target these types of tasks specifically. Scatter plots are among the most basic methods of visually detecting correlation between two variables: if the data points cluster into a linear (or more complex) form, a correlation likely exists. Interactivity can enhance their analytic power, as in the commercial Spotfire tool[8]. Spotfire includes both 2D and 3D scatter plots (among other tools), with configurable plot point attributes and interactive links between data views.

But with increasing numbers of variables in a dataset comes decreasing chances of related variables being shown on scatter plot axes. Parallel coordinates [48] show all dimensions on the screen at once, using one parallel line per dimension with a line for each data point touching each dimension at the appropriate point. In return for complexity, the user can use the tool as a "detective" to uncover relationships between any of the dimensions. Parallel sets [12] extend this idea but focus specifically on categorical data (and a rich set of interactive tools). Star plots also use one axis per dimension, but the axes are radial rather than parallel.

Microsoft researchers take a different tack in addressing multi-dimensional hierarchical data [93]. Their *polyarchy* system shows how hierarchies are related to one another in the context of specific data items, rendering them as maximally-pruned trees. It exposes relationships using animated transitions for hierarchy pairs and a 'stack linked' view for viewing multiple hierarchies simultaneously. But their experiments focus more on the animated transitions between hierarchical views than the effectiveness of the system as a whole, and they note that anecdotally the stack linked view does not scale to more than three or four hierarchies.

[8] http://spotfire.tibco.com/

Even considering only this small sampling of work, it is evident that we might address our users' correlate tasks in other ways. It is reasonable to consider whether one of these methods might be more effective than what is essentially comparing treemaps to each other. The polyarchy use case is similar to our environment with respect to hierarchy, but seems to focus on connecting a few data items at a time rather than larger groups of items as in our SERPs or faceted results. Their pruned representation also removes the context of the full hierarchy and is not space-constrained. Parallel coordinates and star plots are better fits for correlate tasks on their own, but are best suited for numerical rather than categorical data as in our case. Parallel sets address this, but ignore hierarchical data that we are specifically targeting.

3.3 Applications

We have applied our ResultMap implementation to two types of digital repository search interfaces: a directed search environment in the form of a Google-like keyword search interface and an exploratory search environment in the form of a faceted browsing system. For the directed search context we augment the search engine results page with a single ResultMap showing the relevant items in context of a pre-defined classification. For the exploratory environment we have initially supplemented the Flamenco faceted metadata framework with ResultMaps and subsequently developed our own faceted framework augmented with ResultMaps

3.3.1 Keyword Search Engine Result Pages (SERPs)

We have applied our ResultMap implementation to a digital library environment in the form of our HCC EDL (see Figure 3.4) and VADL repositories. We have developed a hierarchical topic taxonomy for the Human-centered Computing and Visual

Analytics fields, classified all repository documents within those hierarchies, and use the classifications as the mapped variable. Brushing links different portions of the SERP, and provide several prospective benefits in this context. Representing the entire document space provides context for query results, which is consistent between successive queries. It also facilitates tasks like detecting outliers and clusters within search results by making them visual instead of textual processes. As we have noted, simpler data graphics (e.g., bar graph, scatter plot) can provide some of this function, but fail to capture any hierarchical complexity. We provide a full account of our SERP implementation, its capabilities and evaluation in Chapter 4.

3.3.2 Faceted Navigation

We have noted that interfaces to single-category classifications are essentially the base case of a faceted classification. Our SERP augmentation constitutes such a case, so our work naturally progressed to investigating how ResultMaps might be applied to generalized faceted classifications. Since SERP ResultMaps essentially show a degenerate case of a generalized faceted system, we can apply ResultMaps to a general faceted environment using one ResultMap per facet—what we call *faceted ResultMaps*. Though in principle we can apply ResultMaps to every facet, in practice, it is unwieldy from both a complexity and efficiency standpoint. Because of the differences between a faceted browsing environment and a keyword search engine, a few key characteristics of faceted ResultMaps vary from the SERP implementation:

- SERP RMs are substitutive—that is, the interface transitions from showing one set of results to highlighting another set of search engine hits. In contrast, faceted systems use a successive refinement process in which each selection adds a new constraint to the query, reducing the number of items remaining in the results. Thus, faceted ResultMaps are subtractive—every document in the repository is

highlighted initially, and additional selections from facets can only remove items from the ResultMaps.

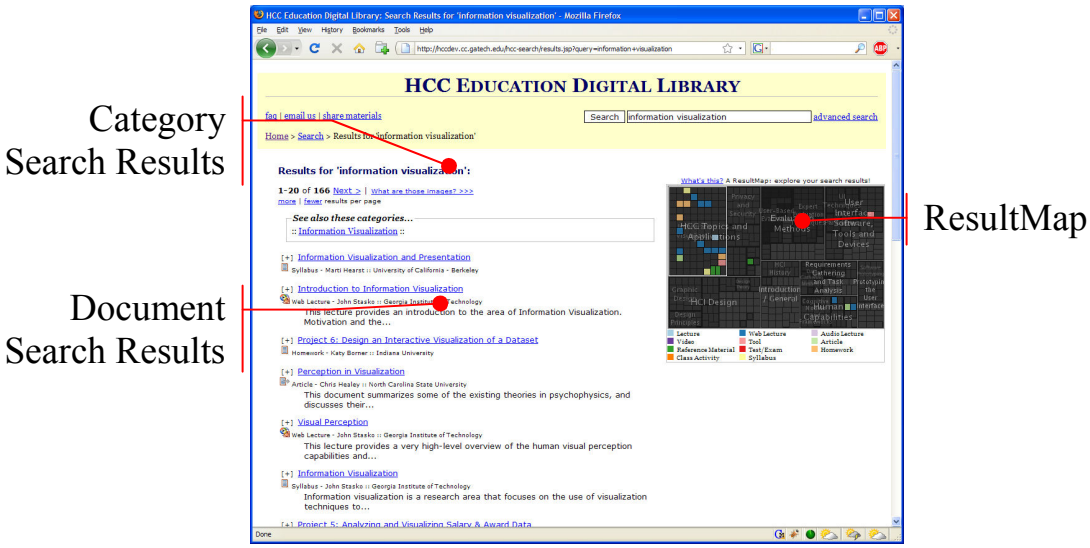

Figure 3.4. ResultMap-augmented HCC Education Digital Library SERP.

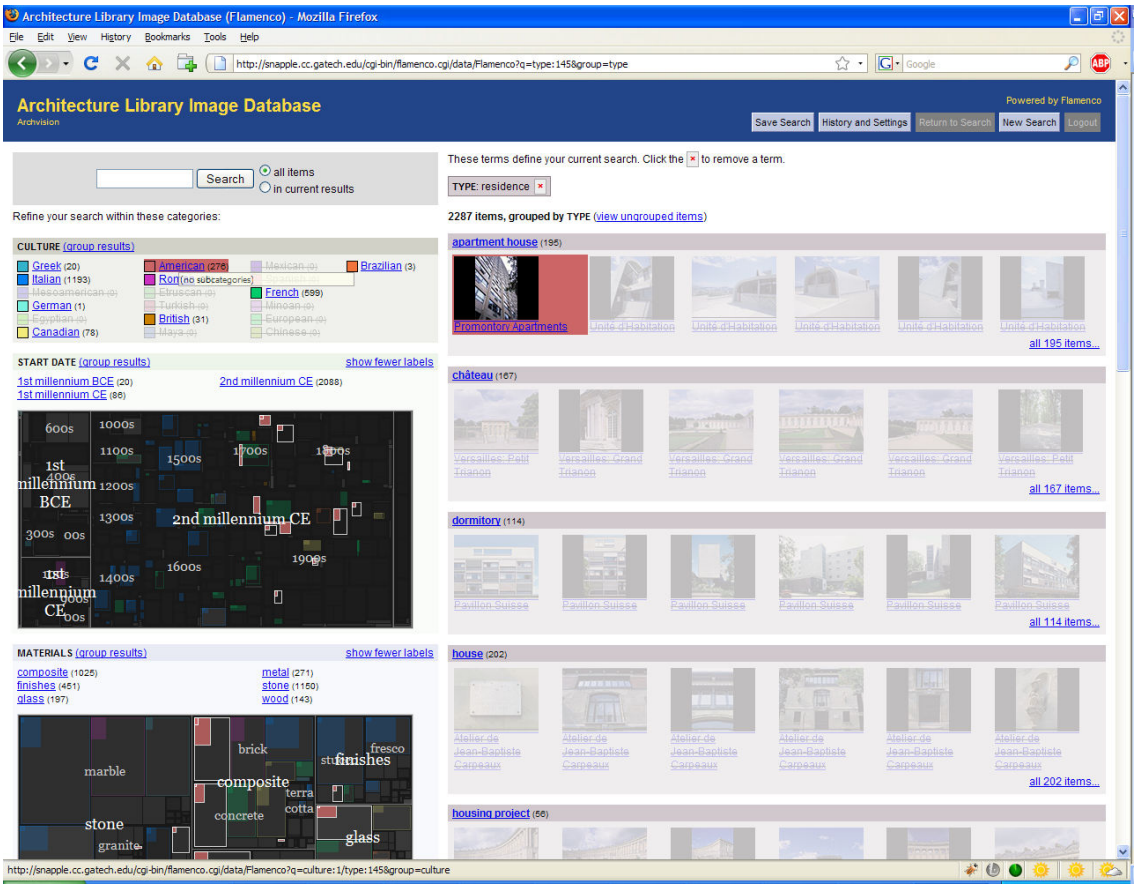

Figure 3.5. A ResultMap-augmented Flamenco instance.

- SERP RMs do not currently provide any awareness of previous or future search results. In contrast, it is important for a faceted system to provide views into both past and future actions.
- SERP RMs show only nodes that are on the current SERP, not the entire result set. In contrast, faceted ResultMaps are only meaningful if they highlight all items remaining in the set.

These differences translate directly into design changes for our faceted ResultMap systems. We initially augmented the Flamenco faceted navigation framework with ResultMaps (see Figure 3.5); based on that experience, we developed the Swivel framework with both ResultMap and progressively-revealed stacked bar chart visualizations (see Figure 3.6). We detail the capabilities of all these versions as well as their evaluation in Chapter 5.

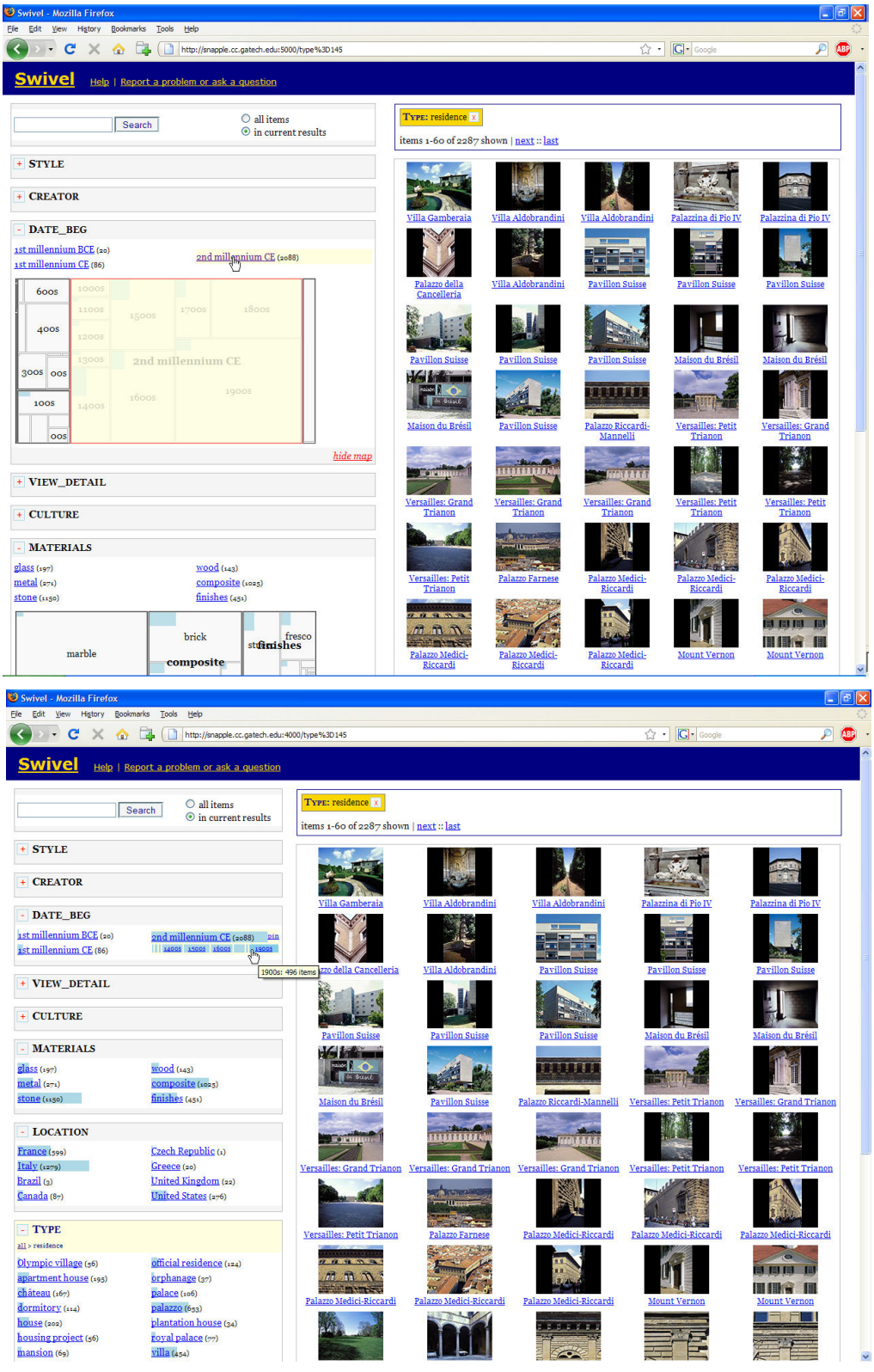

Figure 3.6. The Swivel framework with ResultMaps (top) and stacked bar chart (bottom) visualizations.

CHAPTER 4:

SERP RESULTMAPS

Separately but leading up to this work, we have gathered requirements for, designed and developed two digital repositories: the Human Centered Computing Education Digital Library[9] (HCC EDL) [24, 25, 33] and the Visual Analytics Digital Library (VADL)[10]. These repositories are a contribution to their respective communities by providing free, high-quality resources for teachers, students and practitioners. A side effect of these repositories is that they represent convenient research testbeds. To that end, we have augmented the production search facilities of the HCC EDL and VADL with ResultMaps for both field and lab studies; they are available for general use at the URLs in footnotes 9 and 10. We have used individually-represented ResultMaps in these applications since these repositories' sizes are within the limits of that kind of rendering (cf. Chapter 3, section 3.1.3, p. 44).

As part of the development of these two libraries, we have generated hierarchical topic taxonomies of their respective fields. We leverage those taxonomies to create useful context for the results within the libraries' search engine result pages (SERPs): we use these taxonomies as the structural basis (i.e., the mapped variable) for the ResultMap implementations. The SERP ResultMap visualizes the repository by rendering its contents within a treemap rendering of the hierarchical taxonomy and highlighting the documents that are hits within the current SERP (see Figure 4.1).

[9] http://hcc.cc.gatech.edu
[10] http://vadl.cc.gatech.edu

Moving to the next SERP in a series replaces the ResultMap highlights with those items on the next result page. Documents can appear under multiple topic categories, so there may be more than 10 highlighted nodes. ResultMaps also highlights categories themselves, which our search engine can also return as hits. The ResultMap engine accommodates an arbitrary number of highlights, though we most commonly use 10 or 20 items per page because those are common defaults on general search engines.

Node color denotes document type. We used the ColorBrewer tool[11] to generate a suitable color palette for our 11 distinct document types. A legend just below the ResultMap shows the correspondence between color and document type. We link the ResultMap, its legend and the search result listing via brushing: hovering over a node in the ResultMap highlights the appropriate legend entry, any other corresponding ResultMap nodes and the appropriate entry in the search result listing. Brushing an entry in the search result list highlights the same items as well. In our system, document highlighting entails all of the following (see Figure 4.1 and Figure 4.2):

- ResultMap nodes: outline in red all instances of the document in the ResultMap; emanate an animated yellow box that expands from each ResultMap node before disappearing.
- SERP document list entries: change background to silver.
- ResultMap legend entries: change background to silver; change font color to red; add under- and over-line.

Category highlighting involves (see Figure 4.4):

- Brushing from ResultMap category to SERP category list:
- Outline in red the category in the ResultMap
- Change SERP category list entry background to silver.
- Brushing from SERP category list to ResultMap:
- Outline in red the category in the ResultMap and change its interior to white.

[11] http://www.ColorBrewer.org

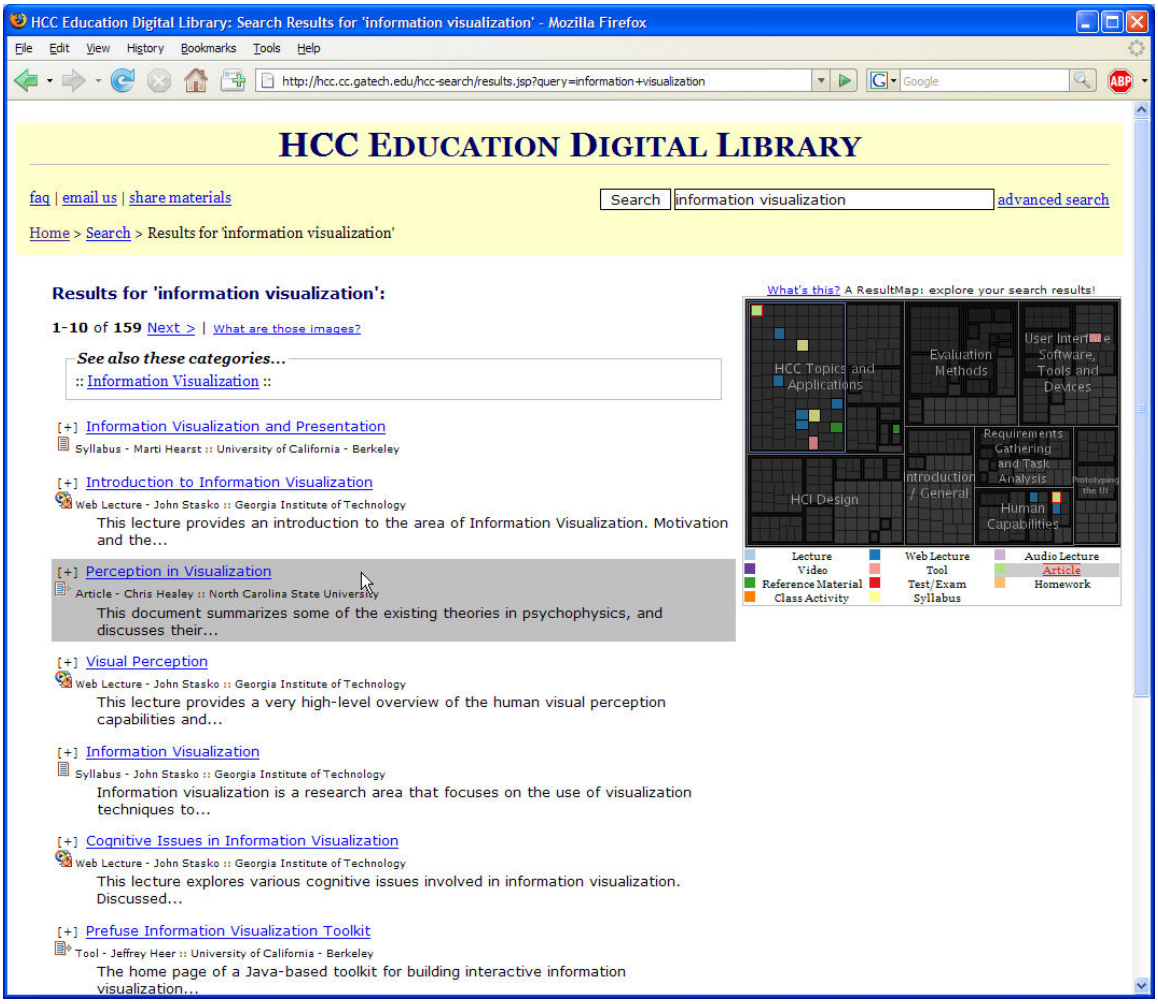

Figure 4.1. Brushing interaction from SERP document list to ResultMap.

Brushing an RM node when the corresponding result listing is outside of the browser viewport shows a message indicating that the user should scroll up or down as necessary (see Figure 4.2). The ResultMap is anchored to the viewport so that it is always at the same position within the window (near the top right of the viewport). Clicking a ResultMap node smoothly scrolls the viewport until the corresponding entry in the list is visible (see Figure 4.3). List entries have both a short and a long form, which contains a full description of the document, if available, as well as a list of its member categories. Clicking the RM node also expands a list entry to its long form.

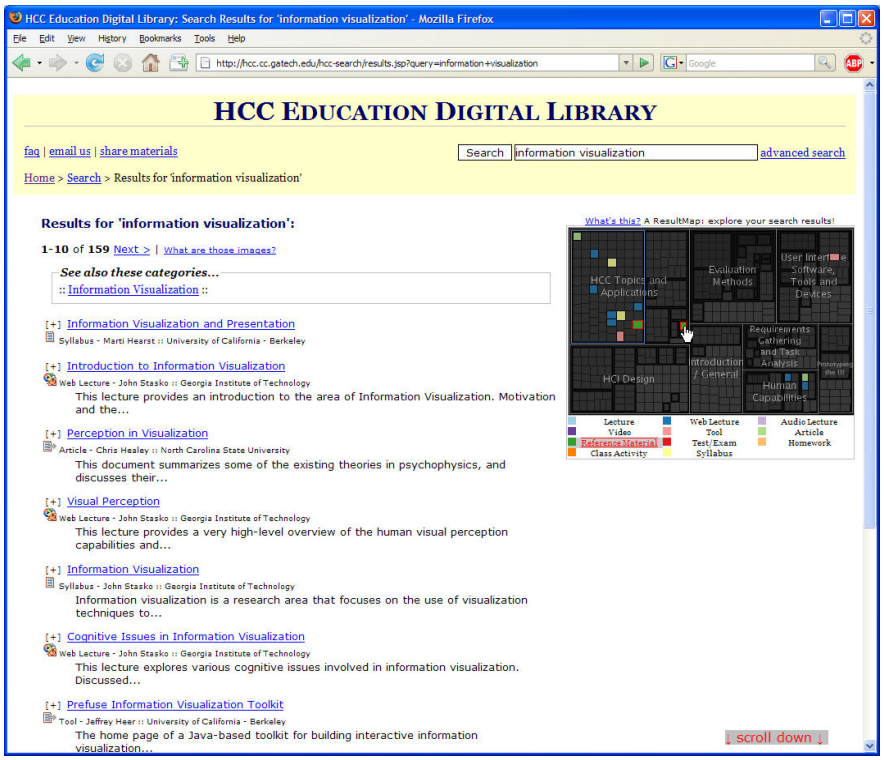

Figure 4.2. Brushing interaction from a ResultMap node to an off-screen item in the SERP list.

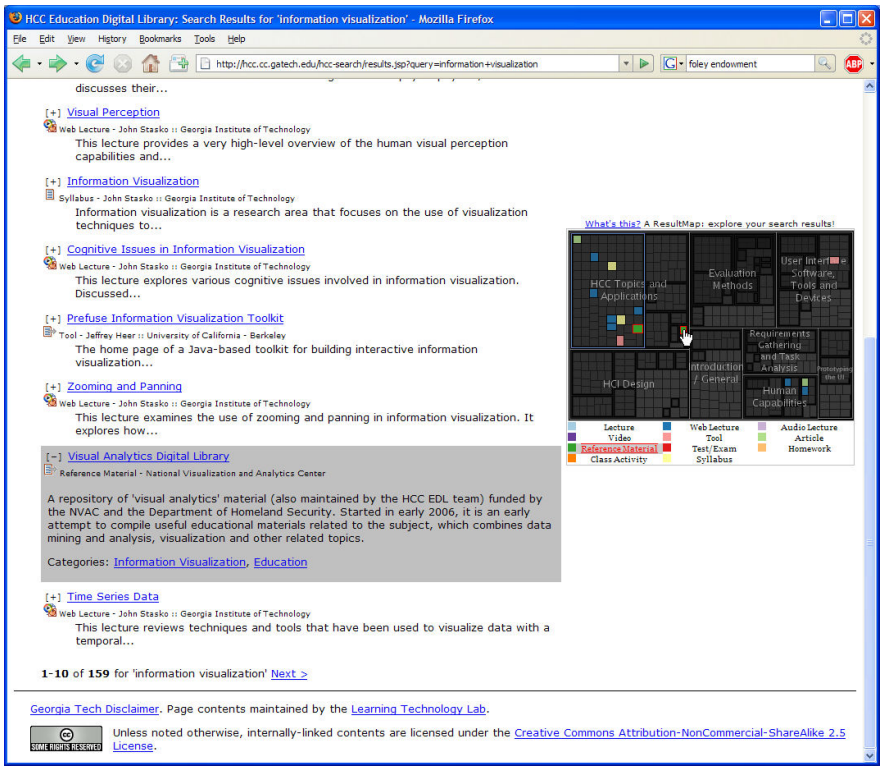

Figure 4.3. The effect of clicking on the node shown in Figure 4.2: the browser viewport scrolls to the item in the document list and expands it to show its full details.

ResultMaps provide several prospective benefits in this context. Representing the entire document space provides context for query results, which is consistent between successive queries. It also facilitates tasks like detecting outliers and clusters within search results by making them visual instead of textual processes. As we have noted, simpler data graphics (e.g., bar graph, scatter plot) can provide some of this function, but fail to capture any hierarchical complexity.

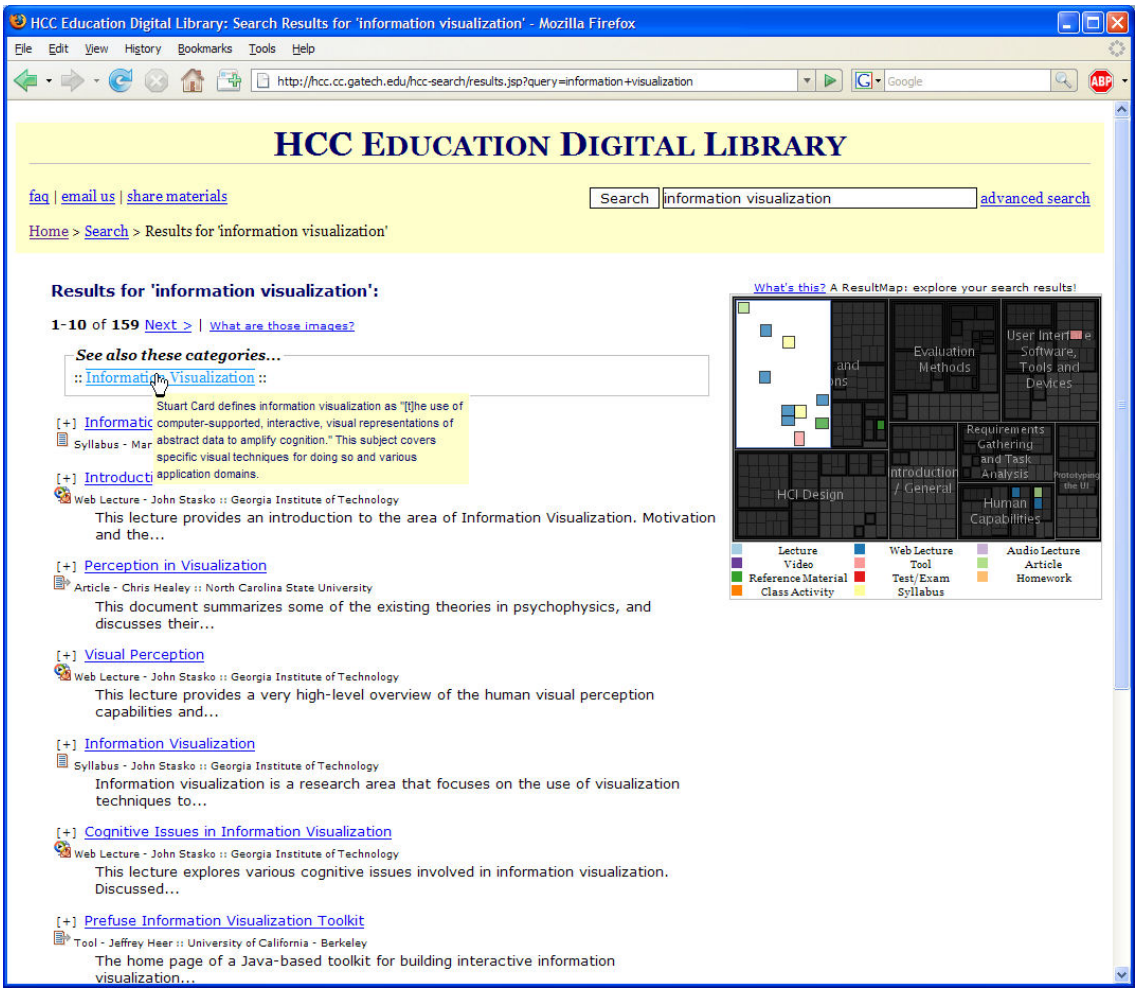

Figure 4.4. Brushing interaction from a SERP category hit to ResultMap node.

4.1 Formative Studies

We conducted several informal formative studies when developing our initial ResultMap implementation. We conducted a set of think-aloud observational sessions in which we asked people to try out a ResultMap-enhanced version of the HCC EDL search engine. Our initial version only linked ResultMap nodes to the text listings and not vice versa; improving the system so that interaction with search results (e.g., mouse-over events) linked to the ResultMap was by far the most common suggestion. We also made changes to a number of typographical elements—font size, justification, etc.—based on feedback from those sessions.

We also conducted an exercise with two sections of an undergraduate HCI course at Georgia Tech (n=35 and n=34). We divided each class into several groups, and asked each group to use the library to find information on various HCI-related topics and prepare a short presentation on them. The groupings corresponded to existing class project teams and the collaborative activity (preparing a lecture) attempted to mimic real-world usage of the HCC EDL and its search results. We gave the classes no direction on whether to use the search engine or normal web browsing behavior, nor did we make any mention of the ResultMap itself or how to interpret it. We hoped to gather log data and qualitative feedback about how the class used the ResultMaps. We did so, but not in the way we initially hoped.

Fewer than 30% of students across both sections even noticed the ResultMap. In fact, our most consistent free-form feedback (9 independent reports) was that the ResultMap resembled an advertisement, causing students simply to overlook it—a finding consistent with Nielsen's warnings about colorful rectangular boxes near the edge

of the screen [78]. The implication for web-based information visualization designers is clear: be wary. But in many cases (including ours) it is hard to avoid using colorful rectangles. In our case, we added a summary text around the ResultMap box explaining its basic purpose and linking to further information. Linking mouse-over events in the result listing to the ResultMap (as suggested by our think-aloud sessions) also has the effect of signaling the user that the ResultMap widget is non-commercial and related to the search results.

4.2 Technical Implementation

The connection between the ResultMap and the rest of the SERP is central to its design. For that reason, we view our use of standard web technology (DHTML) to implement ResultMaps as an important design decision. Building a usable visualization system for the web means conforming to user expectations about the web [76], which implies that ResultMaps should not appear as though they are an obvious layer on top of the rest of the SERP. There are some examples of this phenomenon in academic studies as well. The authors of the Relation Browser system speculated that users had minimal appreciation for its interface features in part because of "the layering effects of the RB over the [web interface] where the primary content is found" [17].

Standalone applications are clearly divorced from the web environment; browser plug-ins or offline applications make more advanced interface features possible, but also make interaction with non-plugin portions of a page cumbersome (Flash) or nearly impossible (Java applets). Since it is precisely such interactions that are important, the additional capability of plugins or external tools is outweighed by the complexity they

introduce—without even considering any ancillary issues like accessibility or search engine indexing.

We use a combination of the prefuse toolkit [44] and Java Server Pages (JSPs) to generate our SERP pages. We cache a background treemap image using the toolkit, and have a site-wide ResultMap object that our search engine JSPs query for each SERP response. The ResultMap object takes a series of document identifiers as inputs and responds with a relatively-positioned HTML `div` element for each highlighted node in the ResultMap. The elements precisely overlay the cached background image at the appropriate coordinates. Cascading Style Sheets (CSS) determine each node's color and JavaScript event handling provides the necessary interactive effects, which are based on element identifying conventions and use elements of the script.aculo.us[12] and Prototype[13] JavaScript libraries. As a result, we leverage web browsers' built-in (and relatively efficient) JavaScript and CSS processing engines.

4.3 Evaluation

We have noted the difficulties of evaluating infovis applications [86]. Here, we outline a series of lab and field user studies with the objective of quantifying and qualifying the effects of SERP ResultMaps on users' information seeking behavior. As we noted in Chapter 2, we attempt to mitigate the difficulties surrounding such evaluations here and in Chapter 5 by taking a multi-pronged approach. Though our overall research approach has a strong empirical bent, we temper it with measures of subjective factors such as engagement, and collect qualitative data in the form of in-study

[12] http://script.aculo.us/
[13] http://www.prototypejs.org/

observation and post-study interviews. Data analysis is primarily a statistical affair on our quantitative data, supplemented by our qualitative data where useful.

We present three studies (R1-R3) that deal with SERP ResultMaps and address research questions 1-4 (see Chapter 5, section 5.3, p. 114 for details of our faceted ResultMap studies).

4.3.1 Study R1: Lab Summative/Formative

We have supposed that ResultMaps assist with detecting outlying results and that rendering the entire repository is beneficial because of consistency. One way in which that benefit might be manifest is the repeated exposure to ResultMaps making users more aware of the full scope of the repository, and in turn giving users a more accurate model of the entire library. This examines these two basic questions about ResultMaps: are there any performance benefits for particular types of search results (i.e., outliers, clustered, etc.) and do ResultMap users get a better understanding of a digital library's characteristics vs. a text-only search interface?

ResultMap nodes can be outliers along two dimensions: color (i.e., document type) or position (i.e., subject classification). We define a *color outlier* as a node with a unique document type (and therefore color) among the current SERP items. To classify nodes as either *position outliers* or within *position clusters*, we give each node within a SERP an outlier score, defined as follows. Given:

- a set of highlighted nodes H
- a set of top-level categories C

Define:

- for $c \in C$ and $h \in H$, $h \in c$ if:
    - h is classified into c or
    - h is classified into a descendant category of c

- a function P: H→ C such that for any $h \in H$:
    - $P(h) = \{c \mid h \in c\}$ (that is, $P(h)$ is the top-level parent category of h)
    - a function N: C→ $\wp$(H) such that for any $c \in C$ and $h \in H$:
    - $N(c) = \{h \in c\}$ (that is, $N(c)$ is the set of all highlighted nodes within *c)*.

We then denote a highlighted node *h*'s *outlier score* as:

$$outlier(h) = \left|N(P(h))\right| \Big/ median\left(\bigcup_{\{\forall o \in \{C - P(h)\} \mid N(o) > 0\}} \left|N(o)\right|\right)$$

Figure 4.5. Outlier score definition.

In plainer language, a node *h*'s outlier score is the total number of highlighted items within its top-level category divided by the median number of nodes in the other populated top-level categories. This score is low when there are few nodes in *h*'s top-level category but many (on median) in other top-level categories. Conversely, it is high when most of the nodes in a ResultMap are in *h*'s own top-level category. We define (*ad hoc*) a node *h* to be an outlier if $outlier(h) < 0.8$ and in a cluster if $outlier(h) > 5$. The middle ground occurs when the numbers of items per populated top-level category are relatively equal—that is, when the result set is spread out and there are neither clusters nor outliers.

In the ResultMap in Figure 4.6, there are 6 top-level categories with highlighted nodes. Clockwise from top left, they have 4, 2, 1, 5, 2 and 2 highlighted nodes. Thus, there is one outlier (blue node far right) with outlier score ½ = 1/median({2,2,2,4,5}), but no clusters since the nodes in the most populous category have outlier scores 2.5 = 5/median({1,2,2,2,4}). There is a cluster in the highlighted category in Figure 4.4 because those nodes have outlier scores of 5.5 = 11/median({1,3}).

Design

We use a single-factor between subjects design with two levels: ResultMap (RM) vs. a control non-ResultMap (non-RM). The dependent measures include task completion time, accuracy, and post-tests of users' knowledge of and subjective attitudes about the DL environment. The RM condition is the search engine interface in use with the HCC EDL, but is an older version than that described above. Specifically, the version used in this study did not brush elements in the ResultMap from the result listings. The non-RM interface removes the RM interface piece, making it similar to a standard search engine (e.g., Google). We make several hypotheses about our treatment effects:

- The RM condition will result in faster and more accurate task performance on outlier detection tasks.
- The RM condition will result in higher self-reported satisfaction scores.
- The RM condition will result in higher scores on assessment of repository knowledge.

Equipment

Subjects used a Windows XP desktop workstation in the Georgia Tech GVU Center usability lab. The workstation was connected to a 1280x1024 pixel LCD monitor and ran a maximized instance of the Mozilla Firefox web browser. Experimental text (instructions, questions, etc.) appeared in a frame 150 pixels high with the remainder taken by the HCC EDL search interface (see Figure 4.6).

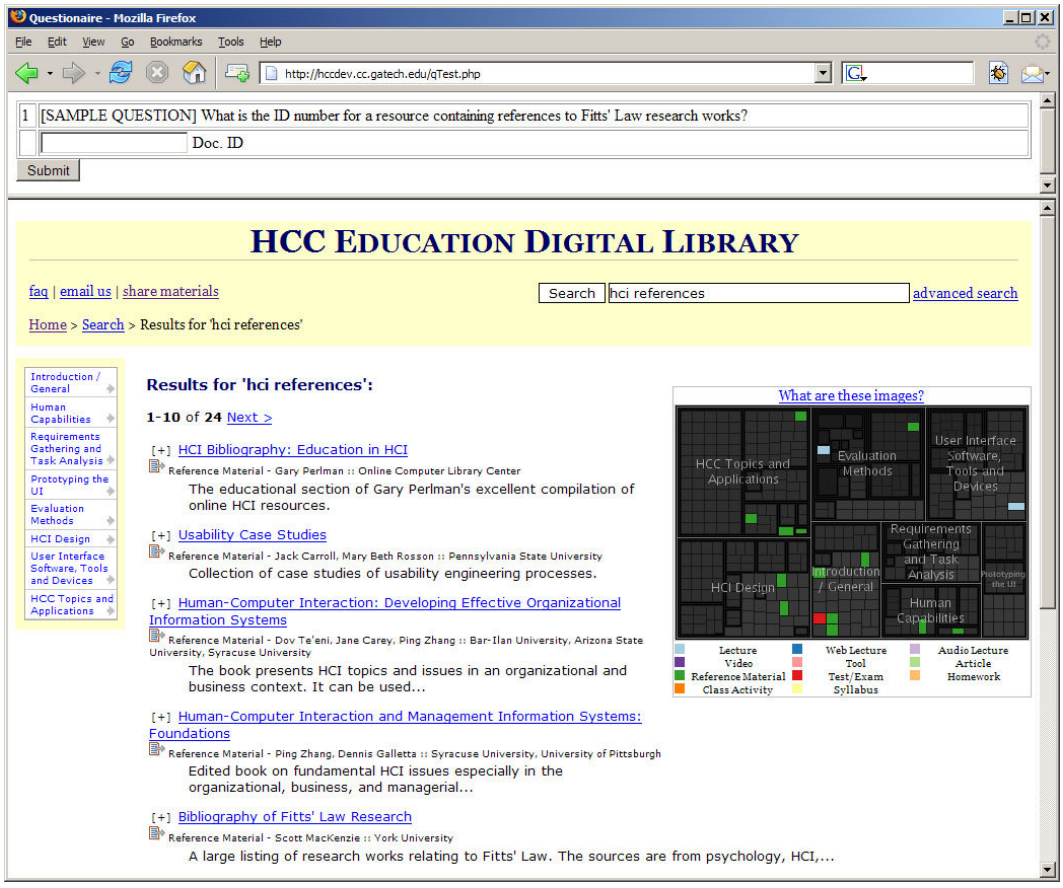

Figure 4.6. The online test environment for studies R1 (and R2).

Procedure

The set of query strings that generated the result listings for the tasks was the same for all subjects to control for search expertise. We informed users what query would be used to generate the upcoming page immediately before showing the listing and accompanied the query with a motivating scenario. For example:

> "The following question relates to the search results for the query '**hci references**'. You might search for this if you were looking for a bibliographic listing of HCI works in the repository."

After clicking a button to continue, the system revealed a SERP (10 items per page) from the HCC EDL for that query string and asked subjects to identify a target document satisfying a set of conditions, such as "[w]hat is the ID number for a resource

containing references to Fitts' Law research works?" All target documents were present on the first page of the search results, although we did not inform subjects of that fact. We instructed participants that document IDs were found on the page linked to by the SERP entries.

We presented 13 search result listings—one practice item followed by 12 randomly-ordered tasks. Prior to the bulk of the experiment, subjects completed a basic demographics questionnaire. After subjects finished the tasks, they completed a set of questions testing their knowledge about characteristics of the repository as a whole (see Appendix B) and about their subjective impressions of the system (see Appendix C). Of the 12 targets, 5 were color outliers, 3 were position outliers and 4 were in position clusters.

Results and Discussion

We completed Study R1 using 20 subjects (10 female) from the graduate student population at Georgia Tech, ages 22-34. All were in HCI-related degree programs with at least a year of HCI experience, which we required because of the nature of the HCC EDL materials. Most (75%) were familiar with the treemap layout technique, but there were no differences attributable to that factor. Participants completed tasks quickly and accurately: a mean of 34 seconds per task with a 90% success rate.

We performed a series of t-tests and a Mann-Whitney non-parametric test on accuracy comparing the RM and control groups:

- There were no significant differences in task time or accuracy between groups for any type of target document; though the RM group had marginally higher accuracy on tasks when the target was an outlier: means of 93% vs. 80% ($p=0.07$).

- RM users rated the impact of the interface on task difficulty more favorably than the control: means of 2.8 vs. 3.9 on 7-point Likert scale with 1 being easier ($t=2.31$; $df=18$; $p<0.05$).
- RM users scored significantly better on portions of the repository knowledge post-test: means of 3.0 vs. 0.2 on a (-6, +6) scoring scale ($t=-3.38$; $df=18$; $p<0.01$).

There were no other significant results. On other parts of the subjective survey, participants were positive about the RM interface design (5-5.5 of 7 depending on question), but no more so than the non-RM group.

We also performed a univariate ANOVA on task time with group and target position as the two factors but found no differences between the RM and control groups. Perhaps unsurprisingly, there was a significant main effect of position within the result list on time ($F=4.37$; $df=7$; $p<0.01$) and a positive linear correlation between the two ($r=0.26$; $p<0.01$).

We take mixed conclusions from these findings. On one hand, we found only marginal support for our hypothesis that SERP ResultMaps improve performance on outlier tasks, and there were fewer survey differences between the groups that we would have hoped. On the other hand, the statistically significant differences we did find all favored ResultMaps—even this older version—and outlier performance trended in favor of ResultMaps. Those differences give some credence to our other two hypotheses that ResultMaps help users understand the gist of a digital library and subjectively prefer them. ResultMaps also perform no worse than the control condition in all cases, indicating that designers can implement ResultMap-augmented search engines without negative effects. We see enough constructive results from this to consider ResultMaps a worthy avenue for additional work.

In reflection, we take several practical lessons from this study. First, the small size of our sample and its between-subjects design made its statistical power relatively low—additional subjects may well have shown a difference between groups in outlier task accuracy, for example. We increase sample sizes and use within-subjects designs where feasible in subsequent studies.

Second, the complexity level of the tasks themselves may not have been high enough to differentiate the two interfaces. Often neither the interactivity required nor the tasks themselves matched well with those identified in Chapter 3, section 3.2 (p. 48). It was surprisingly difficult to generate questions that could only be satisfied by a particular document but also did not just ask for the document by its title, which is a use case for which ResultMaps are not well-suited. Finally, and perhaps most importantly, the deficiencies in the ResultMap version used in this study (i.e., the lack of brushing from the listing to the ResultMap) unnecessarily impaired the RM group.

4.3.2 Study R2: Lab Summative

Our experience with study R1 directly affected our design for study R2. We observed that the complexity of the Study R1 tasks may not have been high enough to differentiate the two interfaces; the tasks may have been too straightforward to warrant ResultMap use, so Study R2 used more complex and open-ended tasks to generate sequences of queries. We posit that ResultMap usage can affect users' query string construction as they make successive queries because ResultMaps make result clusters and outliers more apparent in query results which in turn may feed into future query constructions. The precise nature of any behavior changes is unclear: are users more or

less likely to abandon a query sequence or to start a new one? Are there qualitative differences between queries (e.g., length)?

Our hypotheses about ResultMaps producing better subjective ratings and repository knowledge had better support than task performance, so we focus our dependent measures on those type of metrics. In particular, we add an engagement measure taken from a comparison of online information exploration tools [17], which is in turn a modified instrument from a previous study of interface *flow* in CSCW contexts [35]. We also would like to test ResultMaps against a larger dataset, so we add a second factor to our design. Although we refer to it often as *repository size* and the conditions as *small* and *large*, these labels are partly misnomers: because dataset tasks are inexorably linked to the data, variation due to repository size and variation due to the tasks themselves are conflated. As a result, this design cannot separate repository size effects from those of the tasks themselves.

Design, Equipment and Procedure

We use a split-plot design with two levels of interface type (RM vs. control) as the between-subjects factor and repository size (small vs. large) as the within-subjects factor. We counterbalance the order of the repositories, which serves as a second between-subjects factor. The dependent measures include task time and accuracy, number of query strings, average query length, query string change between successive queries, and our surveys of repository knowledge, subjective ratings and engagement.

We hypothesize that:

- The RM condition will result in task performance no worse than the control.
- The RM condition will result in higher self-reported satisfaction scores.

- The RM condition will result in higher scores on an assessment of repository knowledge.
- The RM condition will result in higher self-reported engagement/enjoyment scores.
- There will be differences in the query string characteristics between the two conditions.

For the small repository condition, we used the HCC EDL; for the large condition we used a portion of the Intute online database, a collection of web resources for education and research. Those materials are categorized according to a detailed taxonomy of topics, and are supplemented with detailed metadata. We extracted the 'Computing' portion of the database for use as our large dataset. The HCC EDL consisted of 583 nodes (487 unique) classified into a 90-node hierarchy. The Intute dataset had 5,518 links (2,602 unique) in 224 categories, and had over 30 document types. To reduce those to fewer than a dozen (for a realistic color palette), we aggregated similar types into broader categories. The physical equipment was the same as in Study R1.

There were a few differences in the software from R1:

- The ResultMaps were a more advanced implementation than in R1, containing all features described above except for brushing from the ResultMap legend to the result list and ResultMap itself.
- ResultMaps in the large condition added a white border to highlighted nodes to improve perceptibility.
- The SERP had 20 items per page by default, and users could add or decrease that number using links at the top of the item listing.
- This change was based on statements from users in R1 (who often stated a desire to see more results on the ResultMap initially) and by a coincidental desire on our part to increase the number of off-screen listing results (that would be shown in the ResultMap).

We created 6 tasks (one practice, 5 test) for each repository ranging in specificity from locating a specific item (e.g., "You are looking for modeling and design software for use in engineering applications. What is the name of a company that supplies such

software?"—Intute dataset) to open-ended directives (e.g., "Find an interesting homework assignment that you would assign if you were an instructor for an undergraduate HCI survey course. Identify its title and author."—HCC EDL dataset). As with the first study, we used search strings as a basis for creating the HCC EDL tasks. We did not have similar data for the Intute library, so we created analogous tasks based on our own knowledge of the repositories. The tasks within each repository were given in the same order; the order of the repository conditions (small and large) was counterbalanced between subjects. As in the first study, the system presented the task and question in a browser frame above the search interface. The systems prompted users to finish after 5 minutes and forced them to move on after 7.

Before the first condition, participants read a short (3/4 page) description of the terminology we commonly use with our experimental datasets (category, document, etc.) and a summary of the type of documents within the HCC EDL and Intute repositories. All participants then watched a 2-minute, narrated screen-captured video about the repository search engines and their basic features (relevance ordering, advanced search features, etc). To control for prior familiarity with the treemap layout technique, members of the RM condition then watched another 3-minute video summarizing the ResultMap features and its interaction with the rest of the SERP.

After the first condition, subjects completed the repository knowledge and satisfaction surveys from Study R1 as well as the IBM Computer System Usability Questionnaire (CSUQ), a validated usability survey [65], and an instrument for assessing task enjoyment and engagement mentioned above (we administered these two instruments in a combined printout; see Appendix A). All surveys used 7-point Likert

scale responses. The repository knowledge questionnaires for each size condition were identical apart from using category names appropriate to the respective datasets in multiple-response questions (Appendix B is the HCC EDL version).

The procedure after the second condition was the same minus the repository knowledge survey—after its first administration, subsequent usage may be skewed towards learning about the repository features relevant to the survey. We also used a survey and informal interview to debrief RM participants about their opinions.

Results

We completed Study R2 using volunteer students from Georgia Tech enrolled in introductory undergraduate HCI and psychology courses. The instructors of the classes offered extra credit to students participating in research studies. Forty-one participants completed the experiment. Five subjects yielded spurious data because of incomplete surveys, tasks, or software errors, resulting in a final N=36 (15 female), 19 in the RM group and 17 in the control. Three members of the RM group had prior familiarity with the treemaps. We graded the task answers on a scale of 0-2 with 0 being completely wrong and 2 being correct. Two graders scored 4 participants independently according to our ordinal grading guidelines; Spearman's rho indicated strong agreement at 0.878.

We performed a series of univariate analyses of variance (ANOVAs) on our dependent measures, which we categorize in Table 4.1 by how the data was collected (behavior records vs. questionnaires/surveys) and the kind of the data collected (subjective vs. objective). Because participants only completed one repository knowledge assessment, we used repository size as a between-subjects factor for that

ANOVA. In our mixed univariate ANOVAs, we include order as a second between-subjects factor since we counterbalanced the order of the repository conditions.

Table 4.1. List of categorized statistical analyses for Study R2.

Behavioral Objective	Mixed repeated measures/between-subjects univariate ANOVAs on task completion times, adjusted task times and accuracy.
	Mixed repeated measures/between-subjects factor ANOVAs on number of query strings per task, mean query string length, and mean change between successive query strings (as measured by the Levenshtein minimum string distance [105])[14].
Surveyed Objective	Two-factor ANOVAs on the repository knowledge indicators.
Surveyed Subjective	Mixed repeated measures/between-subjects factor ANOVAs on CSUQ, enjoyment, and engagement aggregate measures.
	Mixed repeated measures/between-subjects factor ANOVAs on satisfaction indicators.

Behavioral Objective Measures

The task times shown here include both incorrect and partially correct responses, which we judged as appropriate since our tasks had no single correct answer. In any case, analyses which excluded times from wholly or partially correct answers yielded substantially similar results. For our accuracy data, we included partially-correct responses as 50% when aggregating the results.

Table 4.2 shows that RM users had a faster mean task time and a higher accuracy rate than the control group: 12% faster on the small condition and 7% faster on the large.

[14] The Levenshtein algorithm uses dynamic programming to compute the minimum number of character changes necessary to convert one string to another (also known as sequence alignment).

However, the main effect of interface (RM vs. control) was not statistically significant for either measure ($0.11 \leq p \leq 0.34$). Two within-subjects factors were significant: repository size ($F=17.085$; $df=1$; $p<0.01$) and the interaction of repository size and order ($F=12.786$; $df=1$; $p<0.01$), indicating that participants took more time with the larger repository and that this increase was more dramatic when the larger repository was presented first. There were no other statistically significant effects on time or accuracy.

Table 4.2. Task times (in seconds) and accuracy by interface and repository.

	(n=19) ResultMap	(n=17) **Control**
Small (587 nodes)	163s, σ=58s 70.5%, σ=14.4%	183s, σ=71s 59.4%, σ=23.8%
Large (5,518 nodes)	198s, σ= 49s 72.3%, σ=17.8%	212s, σ= 64s 66.5%, σ=24.5%

Likewise, there were no differences in query characteristics (number, length or change) other than due to repository (the larger repository resulted in more, longer, and more varied search queries, all $p<0.05$). In fact, we were surprised at exactly how little interface mattered: within each repository, variation of mean query string length and mean change between successive strings was 2% or less.

Surveyed Objective Measures

The results for our surveyed objective measures—i.e., our survey assessing knowledge about characteristics of each repository—were similar to our behavioral objective measures: RM users generally provided better assessments of repository

characteristics, but no results were statistically significant, including the knowledge measures that were significant in Study R1 ($0.06 \leq p \leq 0.23$).

Surveyed Subjective Measures

On our subjective surveys, RM users consistently gave higher scores than the control group (see Table 4.3). There was a significant main effect of repository size on CSUQ ($F=9.331$; $df=1$; $p<0.01$) and a marginally significant one on enjoyment ($F=4.155$; $df=1$; $p=0.050$), but not on engagement: users thought the larger repository was less usable and enjoyable. We speculate the repository size effect could be due to task difficulty (harder tasks lead to less enjoyment) or aesthetic differences (the larger repository resulted in much smaller RM nodes that are harder to see and select.

Table 4.3. Measures (7-point Likert scales) of usability (CSUQ), enjoyment and engagement by interface and repository. *ENJ significantly different between interfaces.

	ResultMap (n=19)	**Control** (n=17)
Small (587 nodes)	CSUQ: 5.17, σ=0.69 *ENJ: 5.08, σ=1.10 ENG: 5.33, σ=0.88	CSUQ: 4.84, σ=0.83 *ENJ: 4.22, σ=0.97 ENG: 5.00, σ=0.95
Large (5,518 nodes)	CSUQ: 4.61, σ=0.94 *ENJ: 4.91, σ=1.04 ENG: 5.28, σ=1.09	CSUQ: 4.44, σ=0.79 *ENJ: 3.94, σ=0.90 ENG: 5.02, σ=0.77

There were no significant interaction effects. The interface had a significant main effect on enjoyment ($F=8.24$, $df=1$, $p<0.01$) but not on engagement ($p=0.40$) or CSUQ ($p=0.27$). On the satisfaction survey (also given in Study R1), RM users rated their

interface design as making the tasks less difficult and take less time than a standard SERP, but these results were again not significant ($0.09 \leq p \leq 0.12$).

The results of our debriefing survey and interviews reflected an overall appreciation for ResultMaps. The utility of the category grouping and color-coding ResultMap functions were both highly rated (6.11, σ=1.05 and 5.63, σ=1.26 respectively, 7-point Likert scale), and had a mean estimation of using ResultMaps on 65% of the tasks and being helpful 61% of the time. Comments reflected those statistics and centered on their general utility ("The search was awesome"; "The visualization tool was useful") or the functions we identified ("Visual representation of materials in the library is useful"; "It is very easy to see how results are divided into their document types on the map."). ResultMaps also seemed to surpass (perhaps low) expectation: e.g., "The visual map was much more useful than I expected it to be. It was pretty cool!"

Suggestions for improvement yielded several themes: size and legibility; animation; and more brushing. Size and legibility refer the general visualization and labels specifically. Our (purposeful) attention-grabbing brushing animations were divisive: some had an active dislike and some an active preference for the animations. Finally, the inability to highlight all items of a document type from the ResultMap key was a common concern.

4.3.3 Study R3: Field Summative

A naturalistic, quasi-experimental field test was a natural companion to our lab experiments since our lab procedure already used the same HCC EDL search interface as the general public. We modified the HCC EDL in November 2007 so that it showed SERPs with or without ResultMaps. Whether or not to show ResultMaps was pseudo-

randomly determined by the requestors IP address (it was used as a seed to a Java Random object which returned either 0 or 1). Thus, client users randomly received either the control or the ResultMap version of the SERP, and that decision was static for any individual IP. Before and during this time, our ResultMap implementation logged relevant events (clicks, hovering, etc.) by making crafted HTTP GET requests, which were in turn recording via the standard Apache logging mechanism.

The most recent update to the ResultMap engine occurred in April 2008, so we examined the data from then until October 2008. During that time period, there were 22,867 distinct valid requests (ignoring requests from automated web crawlers and malware attacks). Of those requests, there were 516 distinct search requests (272 from RM users and 244 from control users). We extracted from the logs per IP address:

- The number of document requests
- The number of category page requests
- The number of document detail page requests
- The number of list hover events (triggered by users mousing over a list item)

For the RM group, we also extracted:

- The number of RM category hover events (triggered by users mousing over a highlighted category RM node).
- The number of RM document hover events (triggered by users mousing over a highlighted document RM node).
- The number of RM key hover events (triggered by users mousing over an RM legend entry).
- The number of RM click events (triggered by users clicking an RM node, which scrolls to a list item)

In short, we found no statistical differences between the groups, though 31% of RM users interacted with the visualization in some way (as measured by triggering the events listed above).

4.4 Discussion

Like many previous works, the most common theme from our statistical results is 'no significant difference'. However, those significant results we did find favor ResultMaps: positive impact of interface on self-reported task difficulty, repository understanding and self-reported enjoyment. Moreover, there were enough near-significant results (see Table 4.4) that we suspect our statistical power was not sufficient to detect ResultMaps' effect size.

Table 4.4. Marginally significant results from Studies R1 (N=20) and R2 (N=36).

Study	Measure	P-value
1	Outlier Task Accuracy	0.07
2	Task Accuracy	0.11
2	Repository Knowledge (Categories)	0.06
2	Design Impact on Difficulty	0.12
2	Design Impact on Time	0.09

Looking back at our combined list of hypotheses, we find no definitive evidence in favor of any of them, but some mild support for the hypotheses that ResultMaps are subjectively preferred (Study R2) and increase knowledge of repository characteristics (Study R1 and Study R2). The only hypothesis that seems to have strong evidence against it is that about ResultMaps affecting query formulation. The extent to which those measures were identical was striking, and to us a somewhat interesting result.

On a practical level, we have come to distinguish two aspects of users' interest in and motivation towards a dataset: analytic or meta-interest and direct interest in the data itself. DL users, for example, are most often interested in the content of the DL documents rather than their metadata. In contrast, users such as intelligence analysts may

be just as (if not moreso) interested in metadata and distribution thereof around intelligence or police reports. Our evaluations were hampered by the fact that users—even when they did engage with the library data—had more direct than analytic interest, while many of the ResultMap benefits are more analytic. One implication is that ResultMap evaluations might target DL users with more analytic interests: library curators, for example.

Our field study yielded little insight; on reflection, the casual/one-time usage scenario that is common to most general-purpose WWW DLs is not an ideal environment in which to explore usage of a novel addition to a familiar tool (i.e. a standard SERP). Usage which is sustained and more analytic in nature (rather than purely directed information-seeking) better suits the value of our infovis tool. This point applies to our lab studies as well: our hypotheses about opportunistic discovery and insight are more difficult to test in short lab sessions as well. Likewise, the number of near-significant results suggests that the combination of our between-subjects design and relatively small sample sizes proved to be insufficient to detect the magnitude of the RM effect. We consider all of these implications in our evaluation of faceted ResultMaps, which we present in the following chapter.

CHAPTER 5:

FACETED RESULTMAPS

A single-category taxonomy like that used in our SERP implementation is essentially a degenerate version of a faceted classification that has just a single facet. As such, our initial work in SERP ResultMaps progressed naturally to their application the generalized and more complex environment of faceted metadata. In this chapter, we present our work on designing and implementing two ResultMap-augmented faceted navigation frameworks. We revisit our faceted navigation survey from Chapter 2, section 2.4.1 (p. 19) as the genesis of formalizing data and query models that underpin such tools. We developed these conceptual models after work on our first (Flamenco-based) faceted visualization system and concurrent with development of our second (Swivel). We discuss these models first, however, since it provides a useful lens through which we can discuss the design and evaluation of our two systems.

5.1 Formal Models of Faceted Metadata

In section 2.4 (p. 16) we provided an overview of the concepts behind faceted classification and metadata, along with a survey of several modern instances of faceted navigation tools. We broadly organized our Chapter 2 survey around differences between the systems in visual, interaction and structural design. Here, we use that survey to motivate a data-centric look—in the form of entity-relationship (ER) and relational models—at faceted metadata and faceted navigation tools. We first suggest appropriate models for faceted metadata and queries to extract information from them. We then

explore implications these models have on faceted navigation design before presenting our application of the ResultMap concept to a faceted navigation context, which includes implementing two such instances and evaluating their use.

5.1.1 Motivation

A database is useless without effective ways for humans to extract information from it. Among the most prominent user interface (UI) approaches is the SQL standard [75], which takes a programmatic approach via declarative statements. But it is also well-recognized that SQL is not particularly suitable for casual users—the vast majority of computing consumers. Query-by-example (QBE) is more user-friendly in that it does not require substantial syntax knowledge, but it still requires training and some amount of expertise [125]. Another alternative is to sacrifice relational completeness (that is, expressivity equivalent to relational algebra and calculus) for simplicity. Form-based systems take this approach, for instance. Another approach is to enforce constraints on the database schema underlying an application to achieve the desired simplicity.

Faceted classification is one method of constraining a data repository's structure; faceted UIs into such data typically limit the types of data queries available to the users. Faceted browsers thus limit both the structure of the underlying data and the queries into that data, but retains substantial power and end-user ease-of-use [122]. We draw on our earlier survey and design space characterizations to form generalized ER and relational models for faceted browsing systems. Our survey helps to define the boundaries of our data and query models: each should account for the most general forms we have described (e.g., multi-focus over single-focus).

The extent of both data and query limitations imposed by faceted browsing systems has been ill-studied. McGuffin and schraefel have provided graph-theoretic hyperstructure comparisons [74], and Zhang and Marchionini include a BNF grammar for low level faceted UI actions [124], but neither is as useful from a system-level data modeling perspective as ER and relational models. The most closely related work, in fact, is dimensional data modeling (see Ch. 29 in [30]) from the data warehouse topic area (which we discussed briefly in Chapter 2 as well), which utilizes the star and snowflake schema patterns.

These patterns essentially represent an independent rediscovery of faceted classification; as a result the dimensional modeling literature informs this work. Levene and Loizou have presented a model of dimensional classification which is highly relevant [64]. Their work provides a formalization of the snowflake schema similar to our faceted data model derivation, but addresses queries only so far to say that a table can be queried by joining it with its various dimensions and subdimensions. Malinowski and Zimanyi have developed the MultiDim model for data warehouses [70], which focuses on modeling complex hierarchy within dimensions.

The Resource Description Framework (RDF) is a data model for web-oriented metadata and is closely associated with the Semantic Web. Faceted RDF-oriented systems often allow switching the central fact table (i.e., focus data items). Indeed, RDF-based schemas themselves are certainly richly-described data models in their own right.

But despite some similarities, our generalized faceted model differs in key ways from previous work. Like the MultiDim model, we support complex hierarchical structure common in faceted data that are not well-accounted for in traditional

dimensional modeling. But we also account for more network-oriented structures as with many RDF-based systems. But the generality of the RDF approach is such that their creation can be unwieldy and their content polluted with confusing links. For example, the connections in Figure 2.7 on p. 20 rather inscrutably include 'image' and 'article'—while the more obvious Vice President connection has to be manually requested.

The maturity of database solutions means they are both prevalent as back-end systems and have proven to scale to very large sizes and query loads, which is less true for RDF. One exception is the RDF based storage and querying implemented by an ORACLE advanced development team that has been used in bioinformatics applications [22]. We believe that a model presented in database terms is useful and practical. To that end, this section contributes formal data models for faceted schema using ER and relational modeling. We also present a characterization of queries supported by faceted browsing systems using tuple relational calculus. Our system survey both guides and illustrates our characterizations, which we describe now. We use an abbreviated bottom-up look at our data model requirements, discussing how we model basic structures before building a coherent whole. A full explication on the derivation of these models, along with the survey in Chapter 2, is found in a more extensive version of this work [27].

5.1.2 Faceted Data Model

We begin by using the ER model [20] and develop some common schemas that can be used to explain the organization of data in faceted browsers. Given a few simplifying assumptions we can deterministically transform it into a relational model. The primary generalizations for which our model should account—generated from our survey in section 2.4—are hierarchical data, distal facets and multi-focusing. The first is

an intra-entity issue and the latter two are inter-entity. Hierarchical data is self-explanatory. Distal facets refer to facets-of-facets—that is, facets that themselves describe other facets, such as when architectural works have a facet for their architects, and architects in turn have facets for their genders. Multi-focusing refers to the ability of a system to change the focus item set from one entity type to another—for example, shifting the focus of a browser from architectural works to architects.

Entity-Relationship Specification

We make the following simplifying assumptions about the ER model, along with any corresponding relational model implications or choices under each list item:

1. Only binary relationships are allowed.
2. Entities may have a single 1:N intra-entity relationship indicating a hierarchical structure among its members.
    a. Mapping tables/relations are used for all inter-entity relationships, regardless of maximum cardinality.
    b. A single-field, self-referencing foreign key parent models hierarchy; elements without parents have NULL values for the parent attribute.
3. All entities have a single-field primary key attribute id.
4. All entities have an attribute designated as a characteristic string for their members (called name by default).
    a. The name field in entity relation should be NOT NULL.

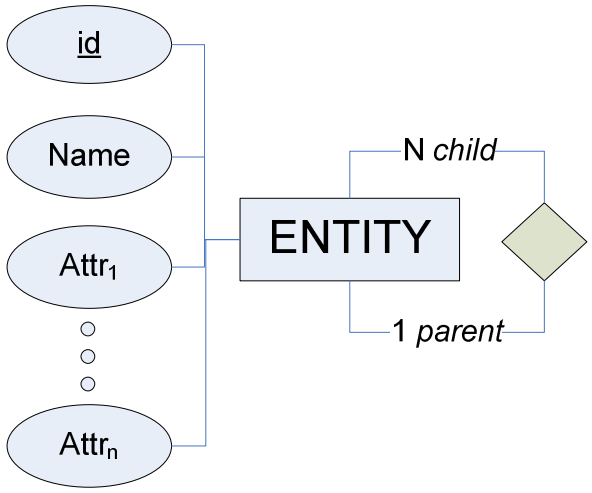

Figure 5.1. Generalized ER diagram of individual entities (*sans* relationships with other entities).

In our modeling, we diagram the most general case possible (e.g., M:N relationships between entity types). Using a 1:N intra-entity relationship handles any tree-like data relationship. In cases where more complex relationships are needed (e.g., multiple parentage), it might be appropriate to split the entity. Figure 5.1 shows an ER diagram of the constraints that relate to a single entity. Nodes without parents (i.e., all nodes in non-hierarchical entities and top-level nodes in hierarchical ones) have NULL parent fields. Henceforth all entities implicitly take this form, but we will not mark the attributes and intra-entity relationship for clarity. Figure 5.2 shows an ER model of a faceted relationship with a single focus entity and set of facets. Note that relationship participation is optional: not all items may be classified into each facet categorization, nor do all facet categories necessarily have item members. Also note that entities may have multiple relationships between them. This is the data model for a single-focus, non-indirect faceted system.

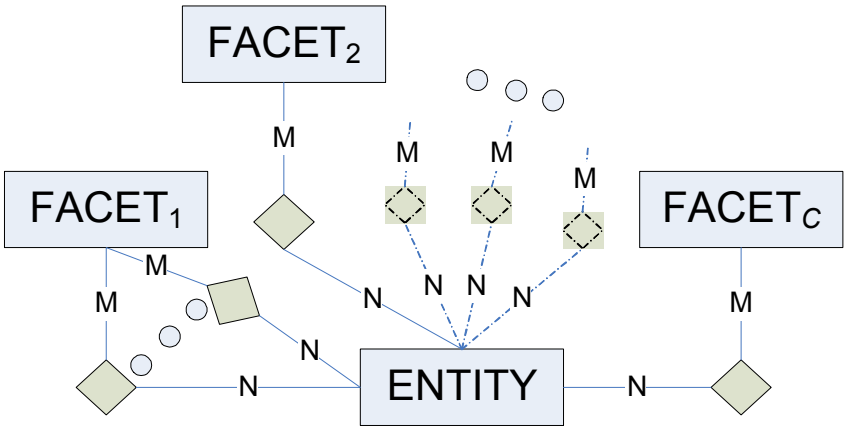

Figure 5.2. Generalized ER model of an entity with *C* facets. All entities and facets implicitly have the form given in Figure 5.1.

In the more general multi-focus, indirect facet case, there is no single root, since there can be multiple foci: rather than an entity tree, the more general model is an entity graph, such as the one shown in Figure 5.3. Any entity might serve in the role of focus entity or as facets for the focus entity depending on the choices made in the user interface. If, for example, the ENTITY4 in Figure 5.3 is in focus, there are 4 facets: ENTITY1, ENTITY3, ENTITY5 and ENTITY6. Alternatively, if ENTITY6 is in focus, ENTITY3, ENTITY4, ENTITY7 and ENTITY8 are used as facets. Note that each of the focus entities can also serve as facets when not in focus and that different focus entities can share facets (here, ENTITY3). Also note that two entities could have multiple relationships between them (not pictured).

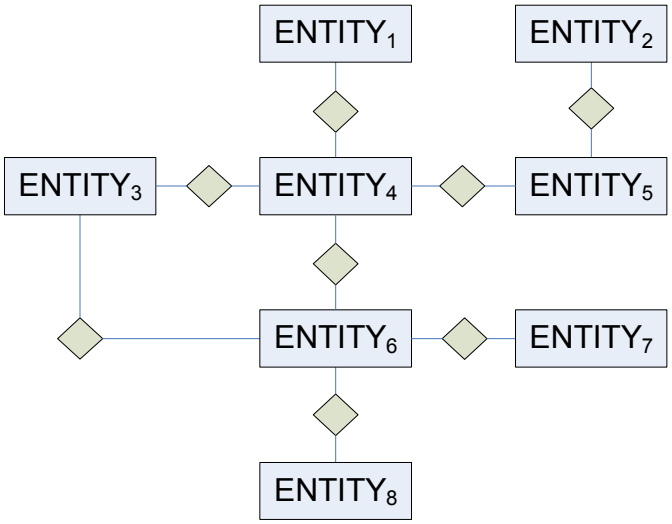

Figure 5.3. An ER model of an 8-entity faceted data graph. Any entity may act as a focus or facet depending on user interest; for example, $ENTITY_4$ would have 4 facets.

Relational Specification

These ER models translate directly to relational schema using standard procedures (see Chapter 7 in [30]) and the rules we list above: exactly one table represents each

entity and similarly, a table represents a relationship. The number of fields in each entity table depends on each entity's specific attributes. Given entities `EXAMPLE1` with *N* attributes and `EXAMPLE2` with *M* attributes, where `EXAMPLE1` and `EXAMPLE2` share a relationship and follow the patterns given in Figure 5.1 and Figure 5.3, we have the relations in Figure 5.4.

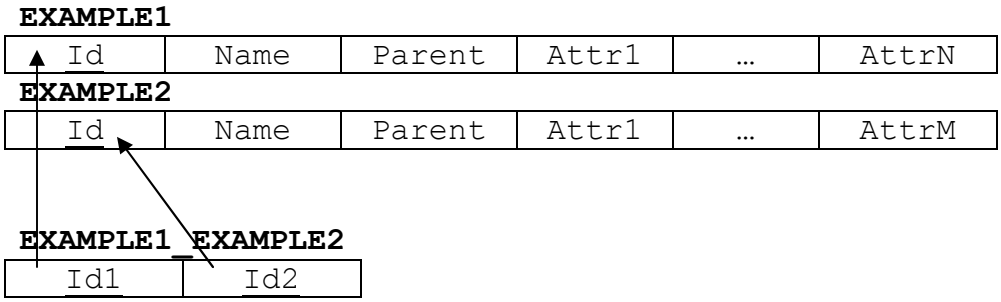

Figure 5.4. Relational diagram (including foreign key constraints) derived from two entities linked by a relationship.

Per our assumptions above, the `Name` fields in the `EXAMPLE1` and `EXAMPLE2` relations are `NOT NULL`. By convention, we assume that the foreign key field `Id1` in the mapping relation refers to the relation closest to the focus item set (i.e., is connected to the focus relation via the smallest number of mapping relations). The third relation in the above design represents a relationship between the `EXAMPLE1` and `EXAMPLE2` entity types—either of which could be a facet of the other.

5.1.3 Faceted Query Model

We have described the structural constraints on the data models supported by faceted systems. There are fewer restrictions on the queries possible against those models inherent to faceted classification. However, we have seen that UIs to faceted data

generally do constrain queries for reasons of simplicity, transparency, and efficiency. For example, faceted UIs commonly expose only facet values which yield populated result sets. Here, we characterize the types of queries necessary to construct a typical faceted UI. Given a set of facet selections, there are two basic types of queries required by the UI: those to acquire the facet values (and possibly item counts), and those to acquire the focus items themselves.

In practice, queries of other forms are needed for user-friendly features. Two examples are breadcrumb selection paths for hierarchical facet selections (*cf*. Flamenco) and previewing facet value selection effects (*cf*. Flamenco tooltips, Relation Brower bar charts). These types of *ad hoc* queries are specific to the type of functionality they enable and so are hard to characterize generically. We therefore focus on the query forms for getting the focus and facet data common to most faceted interfaces, but do so in the most generic form possible.

In the most general case, users may select multiple (potentially negative) constraints from a facet either conjunctively or disjunctively, and may do so for arbitrarily many facets. We must also account for the fact that facets can be shared by different relations and generalize our specification to be focus-agnostic. We assume as inputs the global entity schema, specifically denoted as follow:

- A set of mapping relations E and a relation of interest INTEREST (i.e., either the focus or a facet relation) that follow the relational templates given in the previous section.
- A function GETFACETS: $\{E \cup INTEREST\} \rightarrow \wp(E)$[15] that, given a single input relation *x*, returns a set of mapping relations that are facets of *x*. . If $x \in E$ (thus is a mapping relation connecting two entities), GETFACETS(*x*) are the direct facets of *x* and the indirect facets of INTEREST.
    - If *x* is a mapping relation connecting entities *y* and *z*, GETFACETS returns the facets of the relation closest to INTEREST.

[15] $\wp(S)$ denotes the set of all the subsets of S (its *power set*).

- A set of selection tuples I = $\{(C_1, (E_{1,1},\ldots,E_{1,k_1})),\ldots,(C_n, (E_{n,1},\ldots,E_{n,k_n})\}$ such that:
    - $C_i = \{c_{i,1},\ldots,c_{i,j_i}\}$, where $c_{i,l} \in C_i$ is the l^{th} selection from $\text{FACET}_i \in E$, $1 \leq i \leq n$ and $1 \leq l \leq j_i$. $c_{i,l} \in C_i$ may have a *not* marker, indicating a negative selection.
    - $(E_{i,1},\ldots,E_{i,k_i})$ is the selection path for FACET_i, so that $m = \Sigma(k_i)$ is the number of non-redundant selections and $E_{i,k_i} \in \text{GETFACETS}(E_{i,k_i-1})$ for $k_i > 1$, and $E_{i,1} \in \text{GETFACETS(INTEREST)}$.
- If there is a selection set corresponding to INTEREST, let $C_p = \{c_{p,1},\ldots,c_{p,i}\}$; otherwise $C_p = \{\texttt{NULL}\}$

The C_i selection sets represent disjoint selections from the same facet. The possibility of conjunctive selections from the same facet (*a la* Flamenco) is accounted for by letting different C_i sets to refer to the same member of I, and by allowing the number of constraints n to exceed the number of entities involved in the query. If a C_i selection has a *not* marker, the query form should use an inequality operator rather than equality that we use here for notational convenience. A singleton set is the common single value selection case—for example, selecting only *Gothic* from an architectural style facet.

There is potential ambiguity for indirect facets that are connected to the interest entity in multiple ways. We resolve this by assuming as inputs both the selection criteria (C_i) and the selection path ($E_{i,1},\ldots,E_{i,k_i}$) in the structure of the selection tuples in I. The C_p selection set denotes the special-case set from the INTEREST entity: if INTEREST is not hierarchical or there have been no selections from it, then it should contain NULL. This is necessary since all entities have a parent field, which is NULL in those cases. Conversely, if there is a selection set that corresponds to the INTEREST entity, C_p should denote that set.

The generalized faceted query form is given by Figure 5.5, which requires an (m+1)-way join. When constructing a faceted UI a system would substitute the focus relation for INTEREST when getting focus item data and the appropriate facet relation

for INTEREST when getting relevant facet values. For simplicity, our query form only returns a list of primary key identifiers; in practice, one would include other salient fields from `INTEREST`, (e.g., `t.Name`). There is no standard calculus form for expressing aggregate functions or `GROUP BY`-style SQL clauses, which would be necessary to also retrieve item counts. However, the straightforward SQL translation of our query form may similarly be extended by including a `COUNT(*)` aggregate function and a `GROUP BY t.Id` clause.

$$
Q:\left\{ t.Id \;\middle|\; \begin{array}{l} INTEREST(t) \wedge \\ \left(\begin{array}{l} (\exists e_{1,1}) \cdots (\exists e_{1,k_1}) \cdots (\exists e_{n,1}) \cdots (\exists e_{n,k_n}) \\ \left(\begin{array}{r} FACET_1(e_{1,1}) \wedge \cdots \wedge FACET_{k_1}(e_{1,k_1}) \\ \wedge \cdots \wedge \\ FACET_{m-k_n}(e_{n,1}) \wedge \cdots \wedge FACET_m(e_{n,k_n}) \wedge \\ (t.parent = c_{p,1} \vee \cdots \vee t.parent = c_{p,j_p}) \wedge \\ t.Id = e_{1,1}.Id1 \wedge e_{1,1}.Id2 = e_{1,2}.Id1 \wedge \cdots \wedge e_{1,k_1-1}.Id2 = e_{1,k_1}.Id1 \wedge \\ (e_{1,k_1}.Id2 = c_{1,1} \vee \cdots \vee e_{1,k_1}.Id2 = c_{1,j_1}) \\ \wedge \cdots \wedge \\ t.Id = e_{n,1}.Id1 \wedge e_{n,1}.Id2 = e_{n,2}.Id1 \wedge \cdots \wedge e_{n,k_n-1}.Id2 = e_{n,k_n}.Id1 \wedge \\ (e_{n,k_1}.Id2 = c_{n,1} \vee \cdots \vee e_{n,k_1}.Id2 = c_{n,j_n}) \end{array} \right) \end{array} \right) \end{array} \right\}
$$

Figure 5.5. General form of the faceted query model. INTEREST denotes the entity of interest for this query (which might be the focus or a facet entity for a system gathering data to construct a UI).

5.1.4 Implications of Models for Faceted Navigation and Visualization

The details of our query model are unimportant for the purposes of our broader research into faceted infovis systems. What is important is its ability to broadly characterize the query requirements demanded by faceted UIs and their enhancements.

Our query model shows that even a basic faceted configuration (no indirect facets or focus switching) with *m* facets and *n* selections requires (*n*+1)-way join queries for each facet and the focus items, for *m*+1 total queries. Thus, while additional selections decrease the join selectivity of each query, they actually increase the number of required joins for each of the *m*+1 queries. The consequence is that selections which do not significantly reduce the number of items in the result set may actually increase the amount of time it takes to build the UI because of the increased join complexity. Users can search hierarchical data more effectively when the underlying tree structure is balanced than when it is unbalanced: the model reinforces this guideline. Users have more opportunity with unbalanced hierarchies to make selections that are both less helpful to their goal and potentially detrimental to overall system response time.

Our model also indicates the marginal complexity of various UI features common to faceted systems, such as query preview information. As we noted in the previous section, simple item count previews are relatively straightforward additions (`COUNT(*)/GROUP BY` clauses) to existing required queries. However, more complex data can be considerably more burdensome. Consider Flamenco's tooltip previews, which show sub-category names for hierarchical facet values. The tooltips are generated as part of the overall HTML page UI rendering, so occur synchronously with the rest of the page generation. Assume we have a faceted dataset with *m* hierarchical facets with average branching factor b and *n* selections. Tooltip previews increase the number of queries required to generate a single page by a factor of *b*+1: there are on average *m***b* facet values, each requiring preview data, in addition to the base-case *m*+1 queries. In a

case when there are 9 values per category on average[16], synchronous previews increase the number of required queries by an order of magnitude. Such a dramatic increase makes the most obvious recourse—averting the need for synchronous generation—imperative in large-scale environments.

Along the same lines, we can use the model to assess faceted ResultMaps in terms of both the additional information they provide and requirements they demand. A ResultMap has two major components: the global scope of a hierarchical facet (i.e., the background treemap of all items in the dataset classified into a facet) and the distribution of the highlighted nodes therein. The global scope is constant, so its information gain is approximately the same across different navigation contexts and its cost can be considered constant since its result can easily be cached. Highlighted node distribution obviously varies with user queries, so its gain and cost vary. We consider each case in turn.

The information of the global scope consists essentially of the entire hierarchy tagged with item counts. The query form for this follows our model in Figure 5.5, using an empty selection set and removing the *t.parent* condition that limits the scope of retrieved facet values to a certain hierarchy level. Consequently, this query is invariant to user actions and has results which change only with the underlying data. Assume a base system that shows both global item counts (i.e., number of items matching a category overall) and relative item counts (i.e., number of result set items matching a category given the current query) for categories. Then, the information gain from this query is all the global counts for categories that have a parent other than the current scope.

[16] For example, see http://orange.sims.berkeley.edu/cgi-bin/flamenco.cgi/famuseum/Flamenco

Conversely, the highlighted nodes in a ResultMap show the relative number of items per category and as such vary with user selections. But the highlighted nodes in a ResultMap are leaf nodes and represent categories that are descendants rather that direct children of whatever the current facet scope is—that is, descendants rather that direct children of the appropriate *t.parent* condition from our query form in Figure 5.5. That query is not any more complex join-wise than before, but does require knowledge about all descendants of a category, which might require another query. Alternatively, facet relations could include not just parent but ancestor fields, allowing our query to use conditions over those fields (in place of *t.parent*) directly. In that case, for a faceted dataset with *r* ResultMaps and *n* selections, we require an extra *n*+1-way join query for all *r* ResultMaps.

We have only considered the relatively limited measures of query counts and joins in these rudimentary performance analyses: network latency, dataset size, result set size and are only some of many other factors that determine overall system performance. Nevertheless, this overview show how our data and models can be used to give some understanding of the impacts of UI features and visualization. Furthermore, they are just as capable of facilitating more thorough analyses that include additional factors such as those above.

5.2 Faceted Visualization Design and Implementation

Though in principle we can apply a ResultMap to every facet, in practice such an approach is unwieldy from both a visual complexity and run-time efficiency standpoint. Instead, we target ResultMaps at hierarchical facets—and only ones that are judged most significant by the designer. Because of the differences between a faceted browsing

environment differs and a keyword search engine, a few key characteristics of faceted ResultMaps vary from the SERP implementation:

- SERP RMs are substitutive—that is, the interface transitions from showing one set of results to highlighting another set of search engine hits. In contrast, faceted systems use a successive refinement process in which each selection adds a new constraint to the query, reducing the number of items remaining in the results. Thus, faceted ResultMaps are subtractive—every document in the repository is highlighted initially, and additional selections from facets can only remove items from the ResultMaps.
- SERP RMs do not provide any awareness of previous or future search results. In contrast, it is useful for a faceted system to provide views of both past and potential future actions.
- SERP RMs show only nodes that are on the current SERP, not the entire result set. In contrast, faceted ResultMaps are only meaningful if they highlight all items remaining in the set.
- Unlike the arbitrary topic taxonomies, many faceted classification dimensions have inherent order (e.g., dates or sizes).
- SERP RM color encoding is determined by a key attribute, which is typically a salient metadata feature of the data set. Such data is likely viewed as a facet in that context, so faceted RMs use a particular facet as their key facet.

These differences suggest several useful design changes for faceted ResultMaps:

- Since the initial state of the system includes all items in the dataset, a scalable ResultMap implementation is warranted (i.e., collectively-represented ResultMaps as outlined in section 3.1.3, p. 44).
- ResultMaps should provide order-preserving layouts for ordered metadata.
- Brushing a node in a faceted ResultMap should highlight all corresponding nodes in all other faceted ResultMaps as well as its corresponding entry in the result list. But in the new faceted context, an item can be outside of the current page; in that case we brush the link to the appropriate page and clicking the node in the ResultMap follows that link.
- Items subtracted from the result set by the previous facet value selection are de-emphasized, but remain distinct from other items outside the result set (look-behind).
- Mousing over any facet value selection provides a preview of which items will be removed by that selection. Prospective eliminated nodes are distinct from previously-removed items (look-ahead).
- The key facet should be non-hierarchical and show the specific correspondence between facet value and color.

We now present the implementation details of two ResultMap-augmented faceted navigation tools and how they address these design issues.

5.2.1 Flamenco ResultMaps

Our first faceted ResultMap implementation is based on the open-source Flamenco framework [122]. The basic operation of the Flamenco framework is unchanged from the original, including its general applicability to appropriately-formatted data (we refer the reader to various points in Chapter 2 and to the reference above for specifics on its basic operation). The basic components of the Flamenco interface are the facet boxes—consisting of appropriate facet values—on the left side of the browser window and the list items on the right.

ResultMaps are collectively-rendered (i.e., a leaf node represents all items matching that category and key attribute rather than a single node) and contained within their respective facets' boxes. Because the item list is not relevance-ordered, items on other pages may commonly be of interest and should also be highlighted by the system. We do this via transparency: opaque colored nodes represent the items on the current page matching that RM category and key facet value; semi-transparent nodes are those items meeting the same criteria but are on another page; any gray space represents items filtered out. Node color is according to the values of the key facet: colors are assigned to particular values, so items in a category are effectively grouped by their key facet value. The top two levels of RM nodes in the hierarchy are selectively labeled according to space availability; in cases when labels cause undesirable clutter, the user can show fewer or no labels.

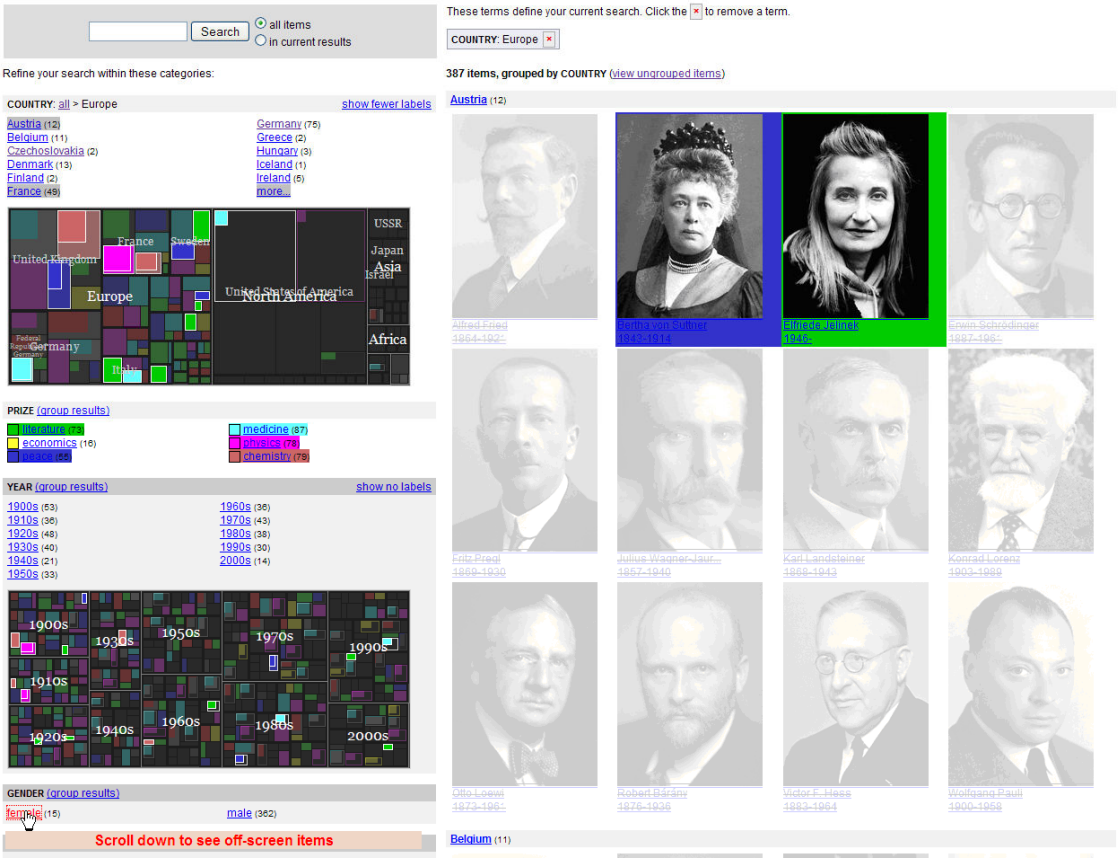

Figure 5.6. A ResultMap-enhanced Flamenco faceted browser showing Nobel Prize data. Brushing links list items, ResultMap nodes, and facet values; brushing facet values previews the result of clicking on any link. Here, the user is hovering over the female facet value (at bottom), which previews the effects of that selection.

Figure 5.6 shows a ResultMap-enhanced Flamenco instance on a dataset of Nobel Prize winners. In it, the user has selected the *Europe* value to show only winners from that continent. *Prize* is the key facet, so node color corresponds to a particular Nobel prize type (e.g., physics, peace, etc.). The facet box acts as a color legend with small colored boxes accompanying each facet values. Filtering on key facet values differs from filtering on other facets: instead of removing all values (which would remove the color key), the other values are crossed out and decreased in brightness.

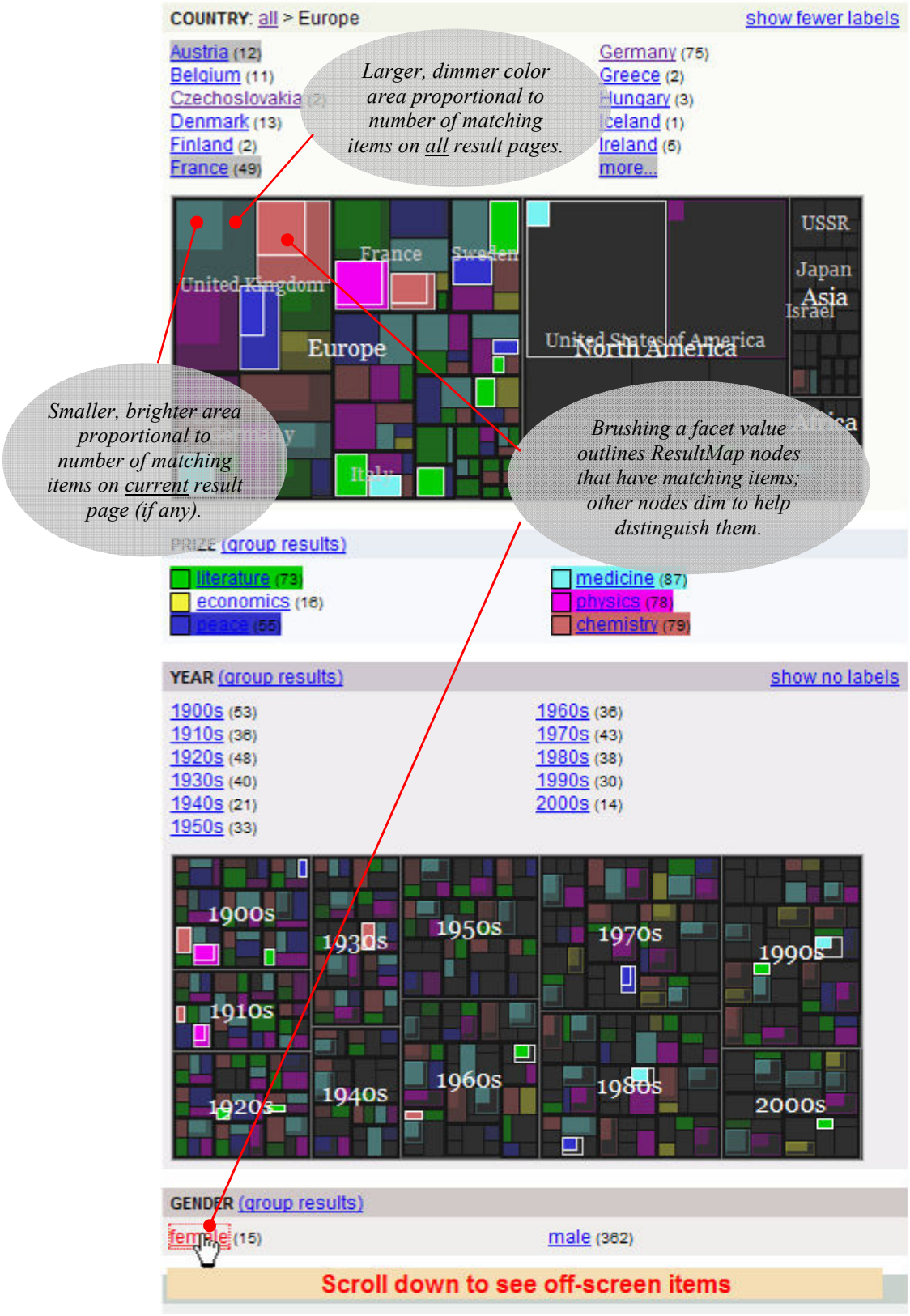

Figure 5.7. Detail of facet boxes from Figure 5.6.

Because the user is looking at European winners, all of the ResultMap nodes under the Europe label have some color to them (see annotated larger detail in Figure 5.7): all European winners appear on some page of the result list. Highlighted nodes have two component areas to them, both in proportion to the number of items matching the current faceted constraints: the larger color area is proportional to the number of items on any result page; the smaller brighter area at the top left of a highlighted node is proportional to the number of matching items on the current page. As such, if there are no matching items on the current page there is a single lower-brightness node; if all matching items are on the current page, there is a single higher-brightness node.

Figure 5.6 and Figure 5.7 also show the effects of brushing any facet value: the system highlights all interface elements—all ResultMaps, facet values, and list items—that match that value; a collective ResultMap node matches a facet value if any of its constituent items match. Thus, brushing is essentially a preview of the effects of clicking on a facet value. The same effects are also triggered by brushing any ResultMap node or any list item. The preview highlights for the respective interface components consist of:

- ResultMap nodes
    - slightly increased color brightness
    - white outline
    - significantly decreased brightness for non-matching RM nodes
- Text facet values links
    - Key facet: background color changes to appropriate color
    - Non-key facet: background color changes to gray
- List items
    - background color changes to matching key facet color or colors.
    - if any list items are out of the browser viewport, a message at the top or bottom of the window indicates that fact to the user.

Clicking on the smaller current-page portion of a highlighted node smoothly scrolls the browser window to the first matching result in the list. The list item highlight

(key facet-colored background as shown in Figure 5.7) fades over 5 seconds to retain context without requiring the user to turn off the highlight manually. Double-clicking any part of a highlighted node acts as a direct filter to that category, bypassing any intermediate higher-level categories in the hierarchy. For example, double-clicking on an RM node in *1911* in the *Year* facet in Figure 5.6/Figure 5.7 would add a constraint for that specific year, bypassing the intermediate *1910s* decade category.

Benefits

ResultMaps in this context provide similar benefits as in the SERP case: making outlier and cluster detection, and a deeper view into facet hierarchies. For example, one might wonder which country has had the most female winners, or what country/prize combination has been the most common. After filtering to only female prize winners in Figure 5.8, the text labels only show that Europe has the most female winners as a continent and peace has the most winners as a prize. But the ResultMap shows that in fact the United States has the most female winners of any one country (the Europe nodes are smaller and spread among many countries, while the U.S. dominates the North American top-level node and has larger individual nodes), and that U.S. laureates in Medicine are the largest country/prize combination (as indicated by the largest highlighted node).

Aside from general outlier/cluster detection, ResultMaps also provide a mechanism to show multiple classification data. In Flamenco, selecting a hierarchical facet value produces new choices that are child values of that selection. In Figure 5.9, the user has selected the Europe value from the *Country* facet, showing child values of Europe (i.e., European countries). As a result, there is no mechanism to show facet

values in the hierarchy that are countries outside of Europe, such as for prize winners that have dual citizenship. But the *Country* ResultMap does show that information, making it apparent that there are several instances of North American and European country dual-citizens.

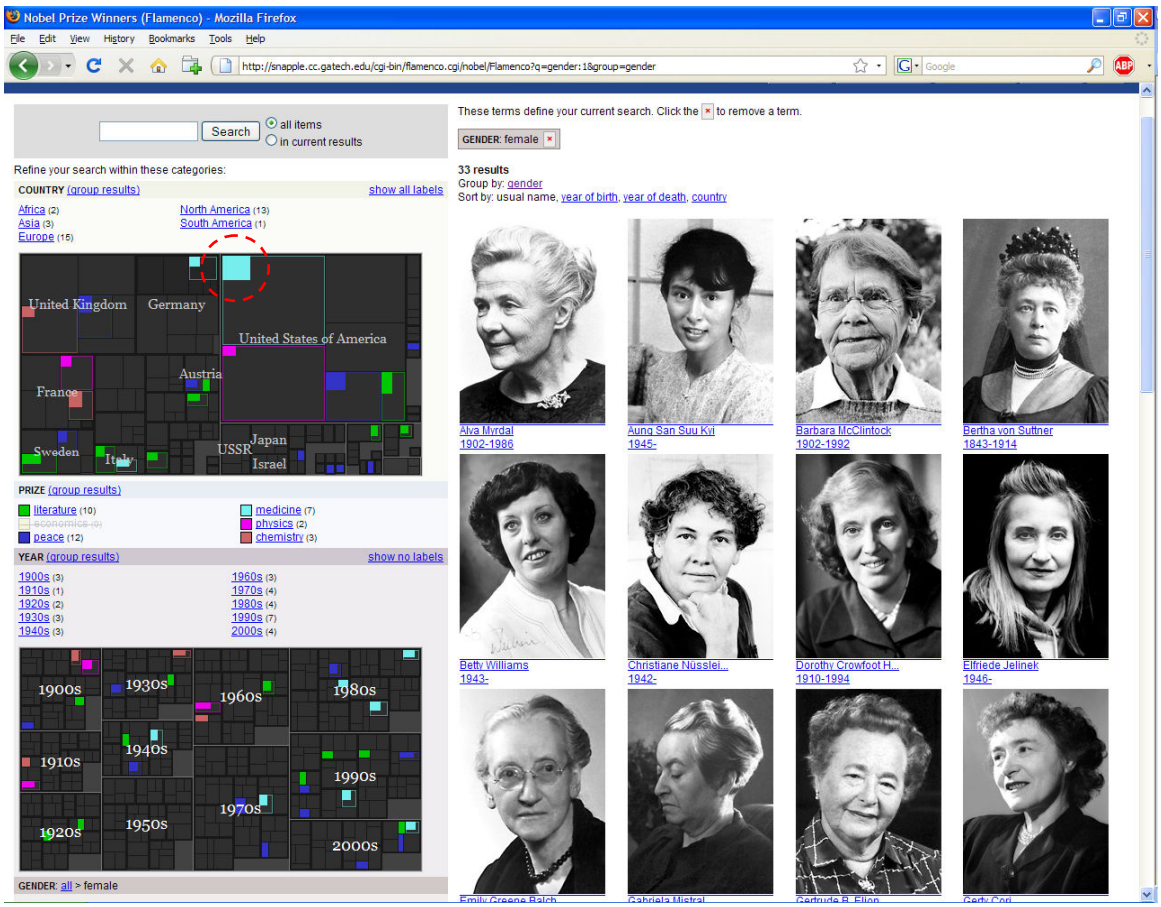

Figure 5.8. Female Nobel Prize winners. The *Country* ResultMap shows U.S. laureates in Medicine are the largest country/prize combination (the largest highlighted node, top center, circled in red).

Faceted ResultMap also allow users to detect more complex data relationships such as correlations between facets. In Figure 5.9, there is no obvious trend in the numeric counts for the decade facet values in the *Year* facet. But in fact, the share of European winners has steadily decreased over time. The raw numeric counts mask this

because the overall number of winners has increased, both through the addition of prizes (Economics) and an increase in splitting the award. The *Year* ResultMap shows this trend: the highlighted nodes become less prevalent from top left to bottom right. The choice to encode the entire repository is crucial here, since it is the juxtaposition of the highlighted with the non-highlighted ResultMap nodes that expose this trend.

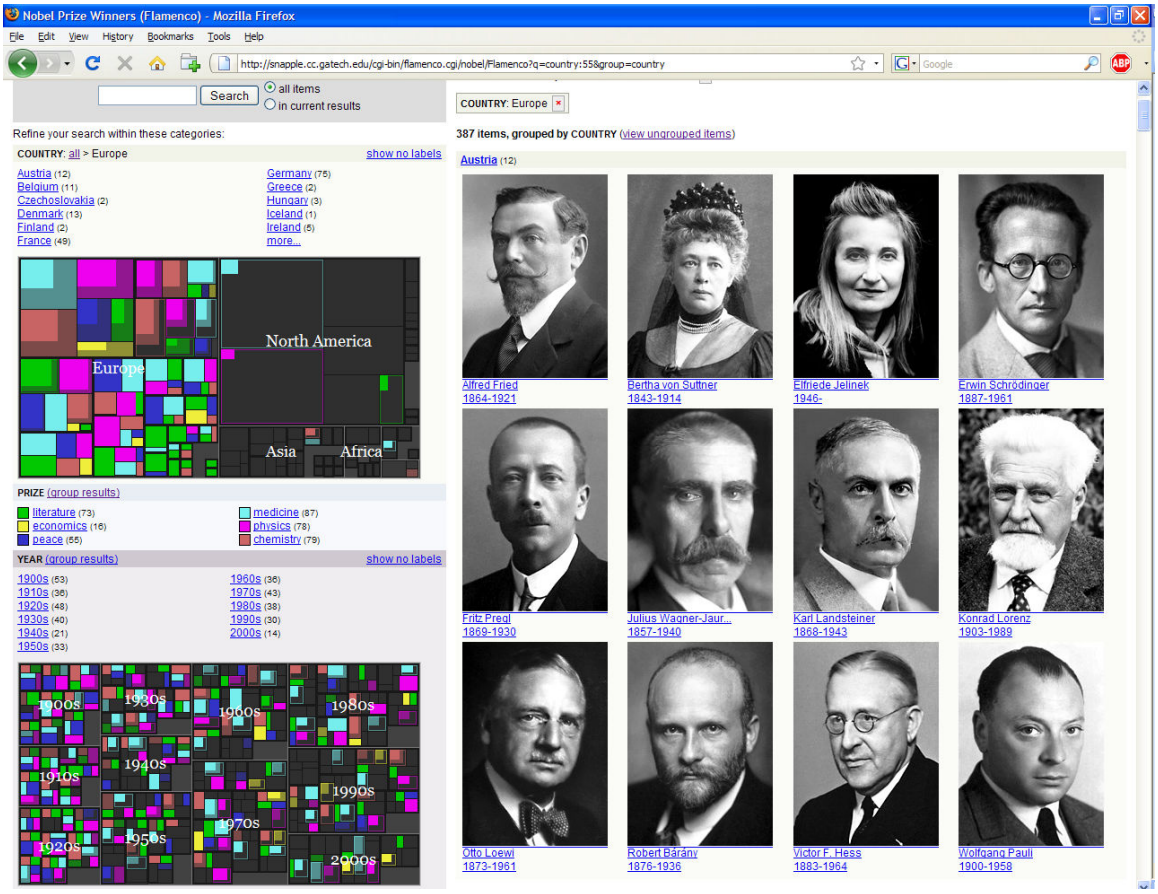

Figure 5.9. European Nobel Prize winners. The ResultMaps show that there is a relationship between continent and time: the share of prizes to European winners has decreased over time.

Query previews can also assist such discoveries by making the effects of facet selections apparent without requiring a click through to a completely new page. Figure 5.10 shows a preview of the Europe value selection that leads to the page in Figure 5.9.

The same general trend in the *Year* ResultMap is present: the lightweight preview interaction means users have more opportunities to detect such correlations.

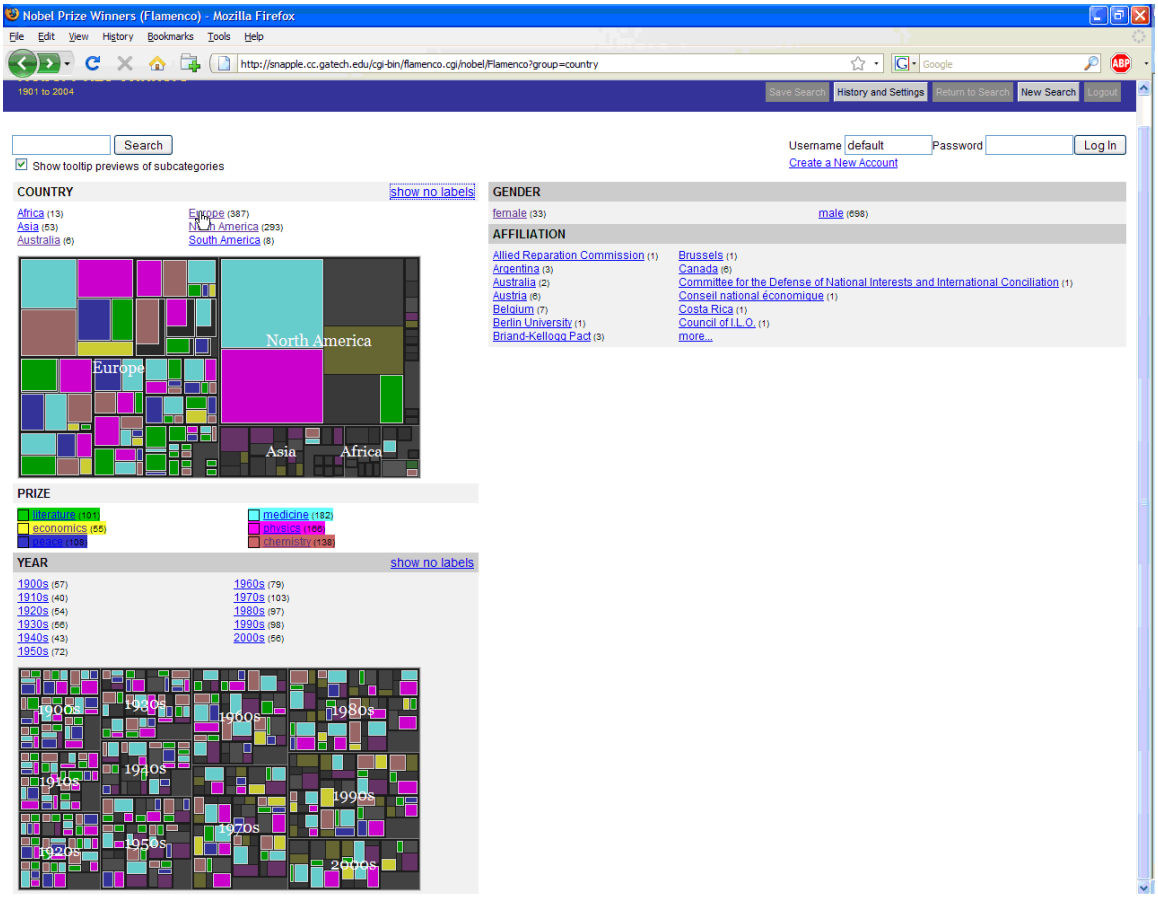

Figure 5.10. Previewing the Europe facet value selection in the *Country* facet. Compare with the result of actually clicking on that link in Figure 5.9.

Technical Design

We modified several aspects of the stock Flamenco distribution: its automated data import script; its MySQL database design; and its Webware-based[17] page generation code. To the data import procedure we added several steps that ask what (if any) facets users want visualized using ResultMaps. If there is more than one, the script further asks

[17] "Webware for Python is a suite of Python packages and tools for developing object-oriented, web-based applications." See http://www.webwareforpython.org/

whether the facet is ordered or not, whether it should be individually or collectively rendered, a static size for the ResultMap, and for a key facet. The script warns the user if the key facet is not flat or if it contains more than 12 values (the size of the largest palette generated by the Color Brewer tool used in our SERP ResultMaps).

Based on that information, the script creates several additional components to the stock Flamenco database design: one additional table per ResultMap; 3 additional fields (storing what facet is key, which have ResultMaps, and whether they're ordered) in a table that stores facet information; and one additional field in the key facet table to store color correspondences. The import script automatically creates key colors by evenly dividing the 360 degree HSV wheel among the values.

The import script then executes a similar prefuse-based Java object as in our SERP ResultMaps. For each ResultMap facet, it queries the database for the facet and item data, performs some data cleaning operations (accounting for multiple classifications both within the facet and within the key facet) and lays out a treemap of the specified size. It then stores the coordinates of all nodes back into the ResultMap-specific MySQL table and writes a background treemap image to disk, as in the SERP implementation. This offline operation must be repeated after any data insertions or deletions.

During operation, the Flamenco server issues a query to the database for the set of items matching the current criteria, limited to a maximum number per page (via an SQL `LIMIT` clause). We add to that query a request for the fields specifying where in the ResultMap an item is classified and build data structures mapping ResultMap nodes to items and *vice versa* (recall we are using collective representation so one ResultMap node

represents many items, and one item may be present in more than one node in a single ResultMap). We then issue an additional query for ResultMap nodes matching any items—that query uses no `LIMIT` clause, but `GROUP BY` limits the size of the result set to at most the number of categories/key facet value combinations. That query includes both relative (via `COUNT(*)`) and global counts (via our offline computed fields).

Given that data, for each ResultMap the system generates a set of relatively-positioned tags for each highlighted ResultMap node. Named CSS styles (the definitions for which are dynamically generated by Flamenco) indicate a node's key facet value, and inline handlers indicate what behaviors (implemented by JavaScript functions) should be triggered for various UI events. The JavaScript function arguments record what other interface components are affected by that node. We use the jQuery JavaScript library[18] to implement many of the UI effects within the JavaScript event handler functions.

We also dynamically generate labels for the ResultMap; by default, we include an HTML label element for every ResultMap node within the top two hierarchy levels containing the category text label. However, in many cases the text overflows a node's bounds. Browser client variation (default fonts, anti-aliasing, etc.) means it is difficult to determine this *a priori*. At page load time, a (client-side) JavaScript function traverses label elements and determines if the text exceeds its bounds by more than a threshold; if so, it tries to decrease the font size until it gives up and eliminates the label.

Problems and Weaknesses

There are several weaknesses with our specific Flamenco implementation and the overall approach we have taken, both from an interactive and a static presentation point

[18] http://jquery.com/

of view. The static ResultMap display is quite dense and often cluttered—especially with labels. The already cluttered display, combined with the interactive complexity of query previews, led us not to implement the look-behind feature we mentioned in the introduction to section 5.2 (p. 95). Scrolling also introduces several problems: if the user scrolls the viewport to see other facets or result items, the ResultMaps may scroll off-screen. Furthermore, if there are enough facets it may be impossible to position all ResultMaps within the browser viewport, making comparisons between different ResultMaps difficult.

On the relatively small Nobel dataset (5 facets, 748 items), the computational costs of our preview data are manageable and page latency is generally within a few seconds. But on a larger dataset of architectural images (10-15 facets, 16,377 items), which we used for the evaluations described in sections 5.3.2 and 5.3.3, latency became unacceptable (commonly 30+ second page load times). In retrospect, this seems inevitable given our analysis in section 5.1.4 (p. 92): there are significant computational costs to previews compounded by Flamenco's synchronous page generation. But our Flamenco-based efforts occurred prior to our work on faceted models and their implications.

5.2.2 Swivel

As a result of our evaluation experience with our Flamenco-based solution (sections 5.3.1 and 5.3.2) and contemporaneous development of our faceted data and query models, we designed and developed our own faceted UI framework, which we call

Swivel[19]. Its basic design is heavily influenced by Flamenco. Figure 5.11 shows a Swivel instance on a dataset of architectural images. As with Flamenco, facet boxes are aligned along the left side of the screen with result set items taking a larger area on the right. The basic click-to-filter operation on facet values is the same as well. The new features over our modified Flamenco include:

- The ability to show or hide a facet box (via '+/-' button at top left).
- Asynchronous ResultMap generation, and the ability to show or hide each ResultMap independently.
- ResultMaps are hidden by default.

Changes from the Flamenco version include:

- There is no key facet.
- We limit the rendered treemap depth to the top two hierarchy levels.
- Highlighted ResultMaps nodes do not have different components for on- vs. off-page results
- There is no brushing interaction between the item listing and ResultMaps or between different ResultMaps.
- There are no previews.
- Clicking (instead of double-clicking) a highlighted node directly adds the appropriate category constraint.

These changes reflect some of the results from our initial evaluation findings, which indicated that we should simplify our Flamenco-based system. Thus, in Figure 5.11 all highlighted ResultMap nodes have the same light blue color (since there is no key facet and no distinction between on- and off-screen result set items) and unhighlighted nodes are a pale gray. Limiting the treemap rendering depth decreases the information provided by the ResultMap, but also decreases the amount of visual clutter.

Removing brushing and preview effects is partially an effort to reduce interaction complexity ('interactive clutter') and partially a concession to performance requirements and time constraints. We retain brushing effects within a ResultMap-enhanced facet, in

[19] Swivel was so named because it was originally designed as a multi-focus faceted browser, which allow users to refocus or *swivel* around different entity types.

which hovering over a ResultMap node highlights the corresponding facet value and *vice versa*. Figure 5.11 shows the user hovering over the *2nd millennium* facet value in the *Date_Beg* facet (beginning date of construction), which draws a red-outlined light yellow box over the corresponding ResultMap node. We retain these visual effects in this case because they are localized around the cursor and require no extra processing.

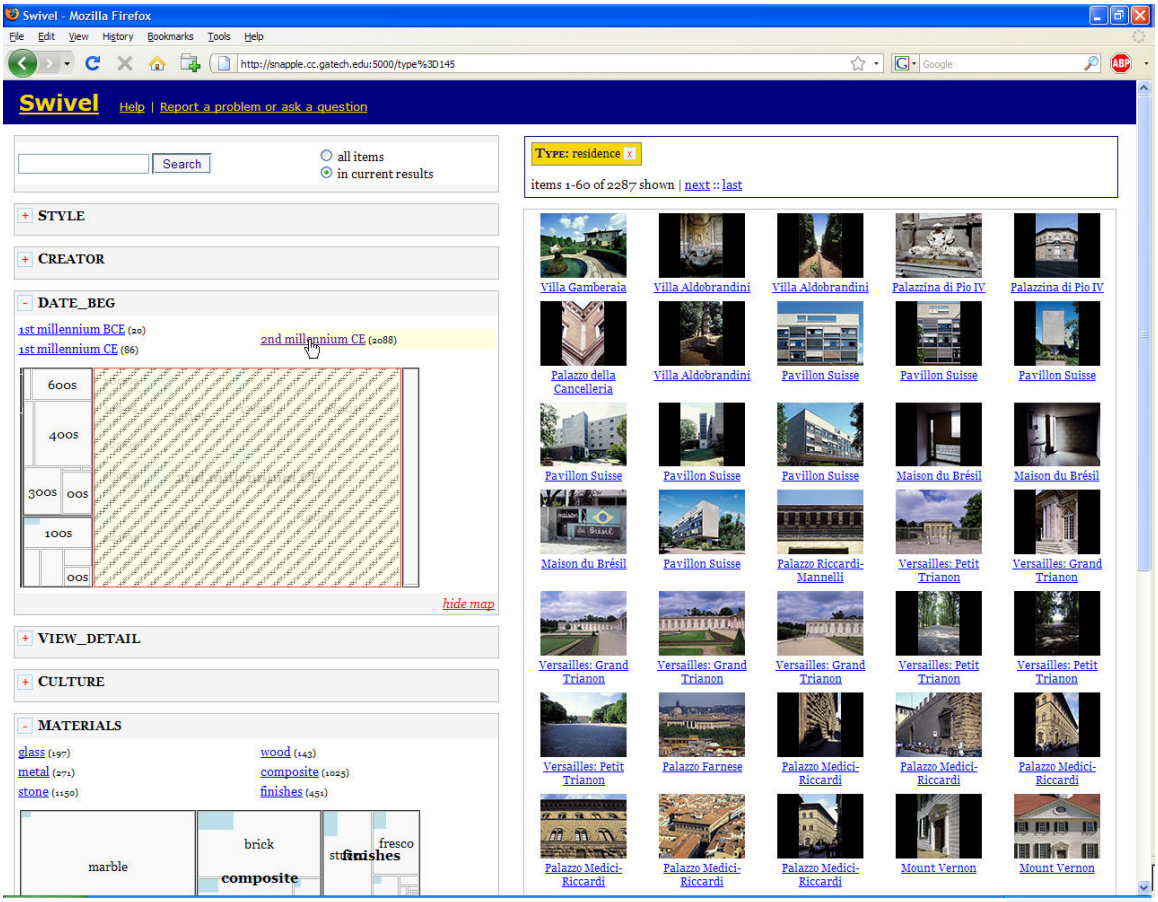

Figure 5.11. The Swivel framework with faceted ResultMaps. The cursor is hovering over the *2nd millennium* value, which highlights the corresponding area in the ResultMap in light yellow (marked with diagonal lines in screenshot for legibility).

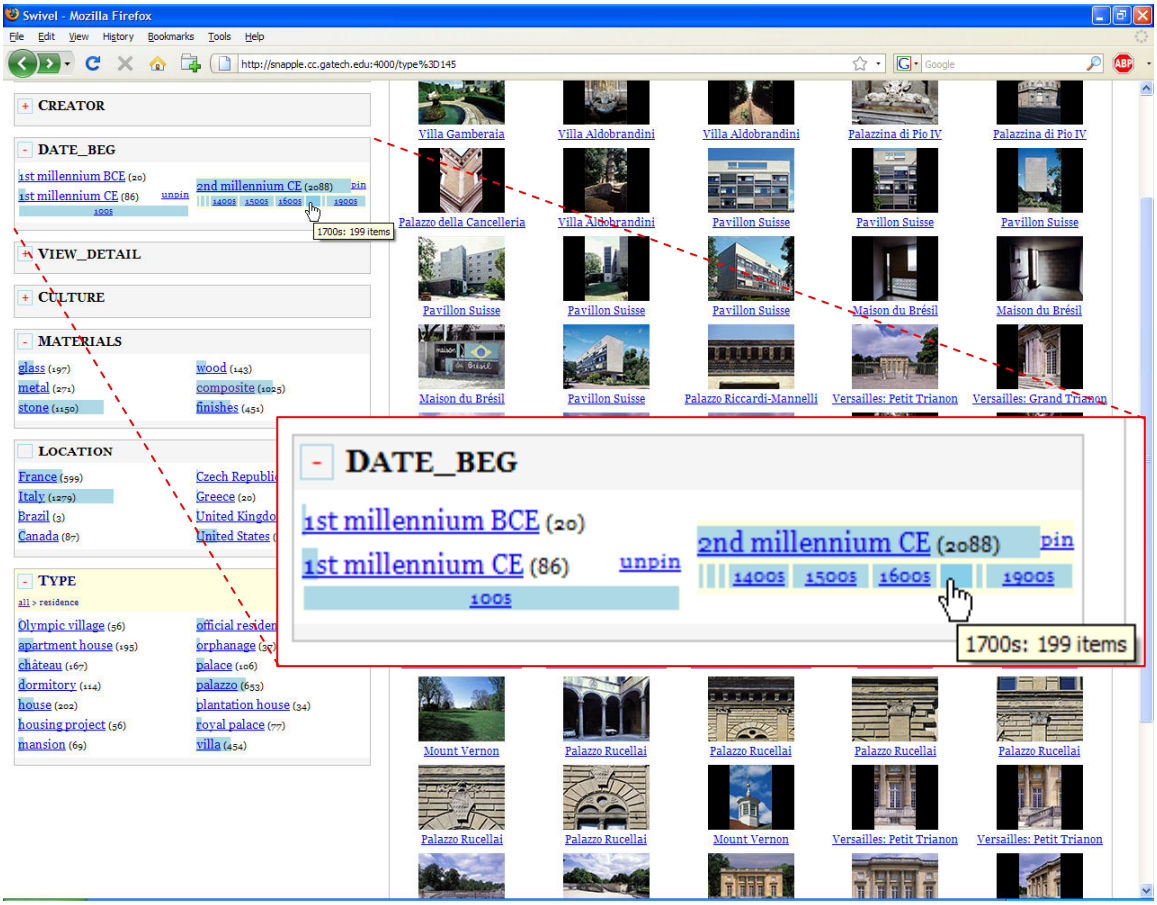

Figure 5.12. Swivel instance with simplified stacked bar-chart version of faceted ResultMaps. The cursor is hovering over the *2nd millennium* value, expanding it to show its sub-categories (region expanded for legibility).

Also as a consequence of our initial evaluations, we began investigating ways in which we could simplify the ResultMap concept further to assist its intelligibility. At the same time, we were interested in how ResultMaps compared to the simple bar-chart graphics from systems like the Relation Browser and Bungee View. The result was a second Swivel version with visualizations consisting of interactive stacked bar charts (bar Swivel in short). Figure 5.12 shows the same data as Figure 5.11 within the bar Swivel version. Facet values have light blue bars in their backgrounds whose length is proportional to the number of items matching that value. A bar that fills 100% of a value's width (the entire column space, not just its text) means that facet value matches

all items in the current result set. In contrast with ResultMaps, values in all facets have an associated visualization component.

Hovering over any facet value expands its area to show similar item count data about its sub-categories. In Figure 5.12, the user has hovered over the *2nd millennium* facet value in the *Date_Beg* facet to dynamically expand its value area, showing a stacked bar chart of its century sub-category values. Like their parent facet values, sub-category bars (*sub-bars*) have lengths proportional to the number of matching items in the result set. Labeling is provided for those sub-categories with large enough bars, but hovering over any sub-bar shows a tooltip with the sub-category name and a count of its matching items. The user can use *pin* link to fix the sub-bar expansion in place (otherwise the facet value area shrinks to its original state). Sub-bar widths are percentages of their parent value's matching items: 100% width of the sub-bar area corresponds to 100% of the number of parent value items, instead of the number of items in the overall result set.

In Figure 5.12, the user is hovering over the unlabelled 1700s century sub-bar, triggering a tooltip indicating there are 199 matching images, and has pinned in place a previous expansion of the 1st millennium value. Both sets of sub-bars fill the entire value width. The 1st millennium, which matches 86 total images, has a single sub-category (the 100s) filling the entire width, indicating that sub-category matches 100% of the 86 1st millennium images. The 2nd millennium, which has 2088 matching items, has many sub-bars that all together comprise 100% width[20]. Thus, the 1700s sub-bar representing 199

[20] In most cases, sub-bars will take up 100% width. However, in cases when items are classified into the parent but not the child categories sub-bars may take up less than 100% width: for example, buildings classified as made of *stone* generally but not *marble* or *sandstone* more specifically. In cases of multiple classification among sub-categories sub-bars may take up more than 100% width: for example, buildings

items is just under 10% of the width. The consequence of this is that the widths of sub-bars from different facet values cannot be compared against each other.

Technical Design

Swivel is implemented using the Pylons web application framework[21] and a MySQL database design that closely follows the data model specified in section 0. That model is relatively close to that used by Flamenco, so the same data is easily ported between the two. We have not implemented a data import tool similar to Flamenco's, so we rely on the Flamenco tool to parse new textual data and run our back-end Java ResultMap code (for our current implementations, we simply copied existing imports of the architectural dataset).

The Pylons framework consists of many useful Python-based components (such as URL routing and HTML templates) which work server-side to generate the end-user web UI. As with our Flamenco framework, we use the jQuery JavaScript library to implement client-side effects. Both ResultMaps and the sub-bar visuals are asynchronously requested via jQuery AJAX calls, so they are only requested for those facets and facet values that the user inspects. We use session cookies to store facet and ResultMap visibility settings for persistence between page changes.

The Pylons framework is a significant upgrade over the Webware-based Flamenco. Using thousands of lines of code (KLOC) as a proxy for system complexity, our entire Swivel codebase (not including the Pylons framework) is slightly over 2 KLOC for the stacked bar version and 2.4 KLOC for the ResultMap version. That

classified as being made of both marble and sandstone. In the latter case, we shrink all bars equally to fit in the available space.

[21] http://pylonshq.com/

compares to well over 5 KLOC for stock Flamenco (excluding components such as indexed searching and logging that Swivel does not implement) and over 7 KLOC for our ResultMap version. Though this is a crude measure, it gives some idea of the relative neatness of the two frameworks to achieve similar results. We have not conducted any formal testing, but anecdotally Swivel performance levels are dramatically improved over the Flamenco solution. We attribute this primarily to eliminating redundant database queries in Swivel and to an improvement in web application frameworks (Flamenco users Webware 0.8.1 vs. 1.0.1 as of February 2009.

Table 5.1. Lines of code (LOC) comparison between Swivel and Flamenco versions.

	Total	Python	JavaScript	CSS	Template
(Bars) **Swivel**	2088	1018	301	417	352
(ResultMap) **Swivel**	2424	1171	382	462	409
(Stock) **Flamenco**	5791	5514	N/A	277	N/A
(ResultMap) **Flamenco**	7214	6439	408	367	N/A

Problems and Weaknesses

Swivel addresses many of the clutter issues found in our Flamenco version, but at the cost of making less data available to the user. The inability to see ResultMaps when scrolling down the item list remains a problem, as does comparing ResultMaps in disparate facet boxes. The latter is mitigated by the ability to hide intervening facets (effectively bringing facets boxes closer to each other), however. A more comprehensive solution could incorporate the ability to pin ResultMaps to the viewport as in our SERP implementation and to reorder facets via drag-and-drop operations.

Our intentional simplification of both the ResultMap and bar versions may be unwelcome by some users. The inability to compare sub-bar lengths between different Pafacet values in simplified Swivel stands out as potentially confusing; the inability to see more than one additional level in the hierarchy is also substantially more limiting than our original ResultMap implementation. Our evaluation (see section 5.3.3, p. 126) attempts to assess the severity of those limitations.

5.3 Evaluation

We have tested our faceted ResultMaps with two primary datasets. The first is a set of Nobel Prize winner data that comes with the stock Flamenco installation. It has 5 facets (affiliation, country, gender, prize and year) and 748 laureates. We modified the country facet data to include continent nodes, making the facet hierarchical. Laureate attributes available to search queries and on item detail pages include name, birth and death years, and links to a photo, a biography and their Nobel lecture.

We conducted initial testing with the Nobel data, but wanted a more rigorous test of our approach. To that end, we collaborated with members of the College of Architecture and the Library and Information Center at Georgia Tech to use the Archivision[22] image collection, a set of 16,377 licensed architectural images and associated metadata. The original dataset contained a large number of metadata fields, which we formatted using simple heuristics to yield 15 facets: creator, creator type, culture, beginning date of construction, end date of construction, form, location, materials, style, subject, technique, type, view detail level, view direction and view

[22] http://www.archivision.com/

source. Other image attributes included a name, alternate title, a description, description source/URL, image details and notes, and photographer and copyright information.

We used both of these datasets in our evaluation of our faceted ResultMap implementations. We conducted three evaluations: a formative think-aloud lab study using the Nobel data and Flamenco ResultMaps (F1); an in-class quasi-experiment using Flamenco ResultMaps and the Archivision dataset; and a longitudinal quasi-experiment using Swivel and the Archivision dataset. In all cases, we were interested in encouraging open-ended usage of the systems and assessing their impact on insight generation, engagement, and satisfaction.

5.3.1 Study F1: Formative Lab Study

With our formative study, we wanted to get an initial sense of not just the strengths and weaknesses of Flamenco ResultMaps, but also a sense of what kinds of tasks and study procedures might prove effective in our summative evaluations. To that end, we conducted a study of 7 users (5 male) using a think-aloud protocol and the Nobel dataset, followed by semi-structured interviews. The Flamenco ResultMaps had query previews (functional because of the small dataset) but did not have the ability to add constraints directly from the ResultMaps (by double-clicking on highlighted nodes). We implemented ResultMaps on the *Year* and *Country* facets, and used *Prize* as the key facet.

Participants were from the Georgia Tech College of Computing population (but were not HCI students) and received a $15 gift card. Three participants had prior familiarity with treemaps via disk usage tools. We introduced faceted metadata generally and the features of the Flamenco system specifically in a 5-minute screencast video,

followed by a 5-minute scripted introduction of the ResultMap features. Four 5-10 minute tasks formed a loose framework for participants' usage:

1. Find a winner of multiple types of Nobel Prizes.
2. Describe in one or two sentences the relationship, if any, between prize winners' country and the year they won their prize.
3. Choose 3-5 descriptive tags for Nobel Laureates as a group; find someone who does not fit those terms.
4. Find a prize winner of interest to you according to whatever criteria you find interesting.

We verbally presented the tasks as "a few questions [we] would like help answering" using a script rather than on a paper form. We also stated that if they noticed anything interesting to feel free to pursue any tangents. We were interested in an oft-stated benefit of infovis: that it helps people ask better questions, or ones they didn't know they had. We hoped that this casual framing would promote more exploration than would a prescribed question-and-answer sheet. The tasks were also designed to not be easily answered directly by the interface features to promote more extensive exploration of the data.

Participants used the system until they completed all three tasks or had exhausted interest in exploring the dataset. We used the CSUQ and engagement surveys from our previous studies and concluded with a 20-30 minute semi-structured interview about the system and the experimental procedure. We coded the think-aloud results and iteratively card-sorted the concepts we found.

The CSUQ and engagement measures were positive: 5.3 (σ=0.8) and 6.0 (σ=0.7), respectively (again, 7-point Likert scales). Three of 7 users successfully answered each of the first two tasks. We drew four conclusions about the system itself and its usage:

- The ResultMaps were used heavily, by all users on all questions.

- There was a bias towards interacting with the country ResultMap (extant in all users), which was closest to the top of the screen.
- ResultMaps led directly to answers of the first two tasks in 4 of the 6 instances.
- There was a clear desire for OR-type queries—i.e., multiple selections per facet (expressed by all users).
- There was some opportunistic discovery during the main test procedure (three instances of data insights, one instance of lengthy tangential data analysis activity).
- The complexity of the system and visualization is nontrivial—all users expressed uncertainty at least once about predicting or diagnosing the results of their actions.

The level of opportunistic discovery was not as high as we hoped, though over half found some type of unexpected data (e.g., "Hey look, Nelson Mandela … I actually didn't know for sure that he won [a Nobel prize]..."), and in half of those cases those insights were directly triggered by ResultMap usage ("I can … look at each of these decades and … see what's lit up in the country facet. I can see the first 5 decades here [that] the USSR didn't get anything.").

There is an interaction of system usage with the test procedure in commenting about the causes of these trends. For example, it is clear that our multiple-prize winner identification task described above drove the desire for multiple facet selections, which would clearly facilitate the task. This kind of interaction relates to the two main conclusions we drew about our test procedures:

- Task/system capability mismatches between goals and the system created "where to start" confusion among all users on the first task.
- Even the loose, informal tasks we presented induced considerable focus on the goal—4 users said they did not pursue (or look for) tangential data because they were focused on completing the task.
- Some amount of guided introduction to ResultMap features is necessary to lessen initial confusion about how the system works.

In the interviews, users all felt that the procedure felt relatively informal and spontaneous (and that our phrasing of the questions contributed to that)—but our experience was that questions soliciting specific pieces of information (in contrast to

more descriptive tasks) were less likely to elicit opportunistic discovery. This observation affects our approach for the latter two faceted studies. On a practical level, we implemented in-ResultMap filtering enabling users to jump directly to lower-level categories represented by highlighted nodes as a result of this study.

5.3.2 Study F2: Formative/Summative In-class Quasi-experiment

We undertook a larger-scale evaluation of faceted ResultMaps in Fall 2008 using the Archivision dataset described above. We used an in-classroom, quasi-experimental design comparing Flamenco ResultMaps to a control Flamenco, and followed up with interviews of selected participants. We used a between-subjects quasi-experimental comparison despite our preferences for within-subjects comparisons because of time limits (50-minute class meetings) and the difficulties in matching tasks on the same dataset between two within-subjects conditions. Of the 15 available facets, we used 11 for the experiment (chosen in consultation with the architecture professor and graduate students): *Culture*, *Start Date*, *Materials*, *Type*, *Creator*, *Creator Type*, *Completion Date*, *Form*, *Location*, *Style/Period* and *Technique*. Underlined facets had ResultMaps, and Culture was the key facet.

Procedure

We conducted this study with approximately 160 students enrolled in ARCH 2111, an architectural history course in the Georgia Tech College of Architecture. The class was divided into 7 preceptorials (recitation sections) based on students' choices during class registration, which met at 10AM, noon or 2pm for 50 minutes. The Flamenco ResultMaps we used for this version used collective representation; users could use previews only on key facet value links. Overall preview performance was

unacceptably slow for the larger Archivision dataset, which mandated its elimination; key facet previews were available only because they require no more processing than generating the basic ResultMap data. Both Flamenco installations ran on a Dell Precision 370 workstation with 1GB RAM and a 3.0 GHz Pentium 4 CPU under RedHat Enterprise Linux AS4.

We worked with the class graduate teaching assistants to create a short in-class assignment and grading criteria. The assignment tasks were:

> 1a. Search for more than two or three distinctive sets of dates, cultures, architectural styles, etc. associated with glass.
> 1b. Point out other significant associations that arise in your search.
> 2. Search for and list the differences between American and Italian civic buildings.

The grading criteria were relatively specific for task 1a, but on 1b and 2 the goal was to assess the students' degree of insight into higher-level concepts about architecture and architectural history. For example, for task 1b, an expected insight was differentiating how glass was employed in French gothic buildings from its use in modernist architecture.

We divided the preceptorials randomly between the two conditions so that all students in a precept were part of the same condition. For the in-class procedure, students filled out consent and demographics survey forms and we presented a 5-10 minute overview of the capabilities of their respective Flamenco version. The overview covered the basics of faceted navigation (in layman's terms) and in the RM condition, a description of the ResultMap layout and associated interactive behavior. Students used their own laptops and were given approximately 15 minutes per task (30 minute total). After the test, students completed the CSUQ and engagement instruments from previous studies. Our hypotheses were similar to our work on SERP ResultMaps:

- The RM condition will result in task performance no worse than the control.
- The RM condition will result in higher self-reported satisfaction scores than the control.
- The RM condition will result in better task assessments than the control.

We used one preceptorial as a pilot for the procedure. The procedure worked well, but uncovered significant software problems. Our research system had never been deployed to more than a few concurrent users: when exposed to 15 simultaneous heavy users, the result was numerous errors[23] requiring a Flamenco restart to resolve [122]. We implemented two workaround solutions for our test: we separated shared MySQLdb connection objects in the base Flamenco codebase, and created multiple Flamenco server configurations so that each student accessed a distinct instance.

Results and Discussion

The day of the last two preceptorial meetings, a systematic network outage rendered the server inaccessible, so those exercises had to be canceled. In the remaining 4 preceptorials, we conducted the in-class exercise with 38 students (18 male) from 19-22 years of age, two with each of the interface conditions (23 control—13 and 10 per precept; 15 RM—8 and 7 per precept). The architecture teaching assistants compiled a simple grading rubric and applied it to each of the three task sections. Task 1a had relatively objective criteria while the latter two (1b and 2) were judged more subjectively on their degree of insightfulness. In all cases, we again used a three-level grading scale with 0 as wrong or not insightful and 2 as correct or very insightful. On the insight-related tasks, 1 represented mildly insightful responses.

[23] We have never definitively diagnosed these problems, but we believe they are related to Flamenco using the MySQLdb python module in a way that is not thread-safe. Yee et al. mention unspecified 'system errors' in 5 of their 32 sessions that required a system restart under single participant usage. See also see http://mysql-python.sourceforge.net/MySQLdb.html#mysqldb.

Four students encountered problems that were resolved with a system restart; we disregarded task responses from 2 of those subjects and the task 1 response from another. But overall the workarounds we introduced were successful—the constant outages as in the pilot section were not evident. However, system performance was not significantly better: page latency ranged as high as 30 seconds in many cases. The network issues that canceled the last two precept sessions may have contributed to these performance problems as well.

Table 5.2. Mean scores on tasks by interface (p-values from Mann-Whitney non-parametric test on ordinal values).

	Task 1a	Task 1b	Task 2
ResultMap	1.36	1.14	1.36
Control	1.00	0.86	1.17
p	0.13	0.37	0.44

Table 5.3. Mean survey scores by interface (p-values, all large, omitted).

	ENJ	**ENG**	**CSUQ**
ResultMap	5.38	5.54	5.53
Control	5.52	5.22	5.43

Summarized tables of our dependent measures are in Table 5.2 and Table 5.3: once again, although the RM group scored better on all measures except for enjoyment, no difference was statistically significant (using a multivariate ANOVA for survey scores and a Mann-Whitney non-parametric test for the task scores). We did find all the task scores to be positively correlated with each other ($0.374 \leq$ Pearson $R^2 \leq 0.402$; all $p <$

0.05), but not with the survey scores, indicating that students' subjective ratings were not significantly affected by their performance on the tasks.

The CSUQ asks for free-form positive and negative aspects of the system; we examined those statements and coded them according to their content. Users provided a median of 1 positive and negative comment each. The major positive theme (26 students) evoked past studies of FC systems: users like both the concept of multiple categorization and the navigational power it affords: (e.g. "The categories are easy to navigate"; "easy to choose [categories] and change your mind").

The major negative theme was system performance (20 students): response time was quite slow. The system performance complaint was actually statistically different between interface groups: the control group included that comment more often than the RM group (Mann-Whitney $U=98.5$, $p<0.05$). However, the number of students per precept is a confounding factor and was higher in the control precepts (13 and 10 vs. 8 and 7) and is a likely cause of this difference).

There were few other minor trends: some students also commented on the dataset itself, complaining that duplicate images of the same building (e.g., from different angles) 'took over' the results set (5 students). Two independently suggested that images be combined in folders to form single entries for each building: this essentially suggests that architectural works should be the focus entity rather than images of them. Three students also desired the ability to make multiple category selections within a hierarchy level. Four students complained about the interface complexity, but commented on non-RM elements ("[there are] too many links to choose from"; "too many links on the interface").

There was qualitative evidence that ResultMaps are a beneficial addition to basic faceted navigation. Three RM users (including one of the students who listed complexity as an interface negative) specifically listed ResultMaps as a positive aspect: for example, "I liked the color coordination and ease of use to make connections between cultures/periods/media/etc."; "I like how the categories are color-coded to match with relative images." Furthermore, 3 students in the control group expressed a specific desire for additional ways to explore relationships between data items, and implicitly referenced contextualization of result set items: "[It is] difficult to draw relationships with other categories [and] difficult to cross-reference material outside the search"; "Side by side comparison of … categories would be useful; more graphics might be nicer." ResultMaps would seem to precisely fit the criteria posed in such quotes.

We supplemented our main experiment with four semi-structured interviews after the in-class sessions (and did so prior to analyzing the experimental data above). For the post-test interviews, we solicited the entire ARCH 2111 enrollment for follow-up interviews, compensated by a $15 gift card per hour. Two interviews were with students in one of the two aborted test sessions, so it was their first experience with the system. We used these sessions to query users more about the task responses they gave during the in-class procedure (and in the new user case, go through the tasks for the first time). We were interested in the kinds of data tasks they used (e.g., identification, comparison, etc.) to complete the tasks.

The students we interviewed were illustrative of the distinction we described in our SERP evaluation discussion (see section 4.4, p. 80) between users who are interested in DL content itself more than its analysis. On task 1a, two primarily examined the

images in result sets to make qualitative visual comparisons (e.g., "[I] compared the Chartres rose window compared to the Notre Dame one to see the [visual] differences.").

Conversely, two of the students answered the same task by looking at more quantitative or high-level trend in the metadata. One of those users particularly well-illustrated the value of the outlier detection ResultMap use case. In answering task 1a, he specifically noticed a particular ResultMap node and used it to filter his result set, saying:

> "Glass in the 1100s, what could that be? Oh, stained glass." *[After asking about his reasons for his choice]* "I thought most of the buildings in this category would be stuff with glass walls… I was interested to see what that was in the 1100s. The 1100s was the biggest block and alone in this area."

Our major finding of the interviews was that ResultMap complexity was a significant barrier to their use in this use context. Though we had provided a brief introduction to their use in the in-class protocol, users had little time for practice or questions in that setting. As a consequence, users focused on completing even these loose tasks, which damped any inclination to explore features they did not immediately understand. The relatively small sizes of the RM interface components were also an issue based on body language (squinting and leaning close to the display) and participant comments.

At the time of this study, we had a prototype version of the Swivel framework; we took the opportunity to also get feedback from users on that system and potential visualization options during the interview sessions as well. We centered our questions about Swivel design on potential visualization design alternatives. All users thought that some simplification of ResultMaps would be beneficial—the number and small size of the RM interface components were a particularly common request for improvement. Similarly, 3 of the 4 users expressed difficulty in interpreting the color values of the

nodes and thought that feature was too complex. All users also thought they would use the ability to hide ResultMaps, but were split evenly about whether they would prefer them hidden or visible by default.

We also showed users paper prototypes of our bar-chart style visualizations, which benefited from substantial comments on their basic design (labeling expectations, show/hide mechanism, interpretation of their lengths relative to item counts, etc.). Users were also split evenly about whether they would prefer a ResultMap or bar-style Swivel instance.

Discussion

This study was marred both by shortcomings in its software (performance issues) and circumstances (lost experimental sessions), which would have marred any statistical results even if they had been evident. As a result, though our intention was a summative evaluation we look at our results in a formative sense.

Both our experimental data and our interviews indicated that students who understood the ResultMap behavior found them useful and benefited from their features. However, their complexity (in addition to an already relatively complex navigation environment) presents a significant barrier to many users. Since those users who more immediately understood ResultMaps' construction reported tangible benefits from their use (in both the experiment and interviews), we determined that simplifying visual components would be a prominent factor in our Swivel development. Hierarchy size/depth and division of nodes by key facet value are the primary factors driving node size down and ResultMap complexity up. In addition to mostly indifferent reaction to coloring by key facet, we also noted that most browsing sessions did not tend to delve

more than a couple levels into any one hierarchy. Given these two factors, we make two significant ResultMap design changes for their Swivel implementation:

- Eliminate coloring by key facet (i.e., nodes are a single color if highlighted and grayscale if not).
- Reduce the number of RM nodes by aggregating nodes more than two levels deep in a hierarchy[24].

Other pragmatic outcomes obviously include settling on the Swivel platform as the software base for our final ResultMap evaluation, which in early testing had significantly better performance even for single users. Users also seemed motivated to use the system on their own (mentioning its use as a study guide, for example), so we decided on a longitudinal deployment for our last evaluation. By not subjecting the system to concentrated load (since it is unlikely all students would use the system at the same time on their own), we also avoid any would-be trouble from performance scaling. We also hoped such a deployment would promote additional insight discovery [79].

5.3.3 Study F3: Summative Longitudinal Quasi-experiment

Our final faceted ResultMap study examined our Swivel implementation, which is described in section 5.2.2 (p. 107). We used the same Archivision data as in Study F2, but based on observed usage we reduced the number of facets to 8 and ResultMaps to 4 (underlined): *Style*, *Creator*, *Date_Beg* (Start Date), *View_Detail*, *Culture*, *Materials*, *Location* and *Type*. The underscores in some of the facet names reflect database naming constraints that were not eliminated in time for the evaluation. Our goal with this quasi-experiment is again to investigate subjective preferences about different faceted visualization approaches and to assess any differences in insight generation. We again

[24] The choice of two levels is based on the characteristics of our architecture dataset; in other circumstances aggregating at a different level (or not at all) might be appropriate. The aggregation level is simply a parameter in the Swivel framework.

worked with College of Architecture faculty and graduate students teaching ARCH 2112, another architectural history course.

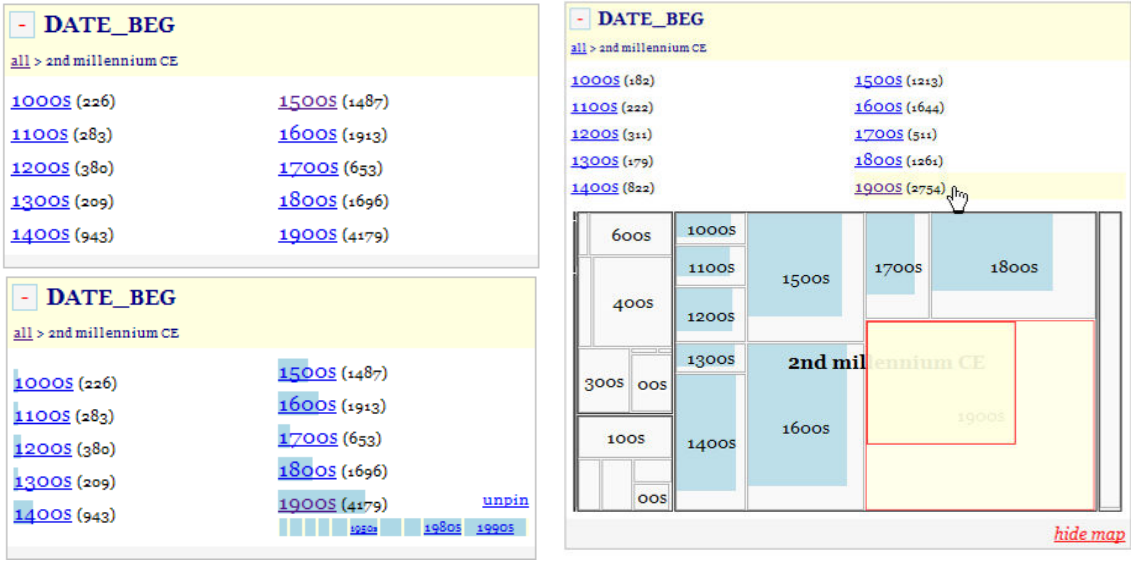

Figure 5.13. Comparison of facet visualizations between the control (top left), Bar (bottom left) and ResultMap (right) Swivel implementations. Each box depicts the same facet under the same constraints in the respective Swivel version.

Design

We employ a within-subjects longitudinal quasi-experiment with 3 levels of visualization interface: ResultMaps (RM), stacked bar charts (Bar), and a control. The only substantial differences between the versions are the facet value visualizations and associated interactive behavior (visual differences within a single facet box shown in Figure 5.13). Because the ResultMaps are statically set to 350 pixels wide, the width of facet column is larger and the item area smaller than the other two versions.

Our previous user studies had been over relatively short periods of time (our SERP field study did not involve any users directly), so we wanted to investigate their usage over a longer period. We also wanted to use longitudinal usage for a more insight-based assessment. We use 3 two-week deployments, one for each Swivel version. Our

standard CSUQ and engagement/enjoyment instrument was the primary dependent measure with a few additions:

- Self-reported estimates of the number of occasions and duration of use of the system during the previous two week period.
- Estimates of the number of occasions and duration of use of the system during the upcoming two week period.
- Self-reports of any insights/discoveries during the previous two-week period
- A post-study questionnaire given at the end of the last two week period (described in more detail below).

Our hypotheses include:

- The RM and bar conditions will result in higher satisfaction and engagement scores than the control.
- The RM and bar conditions will be ranked higher than the control on the post-study questionnaire.
- The RM and bar conditions will generate more insights than the control.

The longitudinal nature of the study means we can deploy all versions of the system to students with sufficient time to use each version, allowing more powerful within-subjects comparisons. The drawbacks to this design are its potential confounds. Most significantly, the ARCH 2112 classroom events are variable over the test period, as are individual differences in other class schedules. The prospect of missing data was also a concern: getting complete data from a student required attendance at four separate class meetings.

Procedure

We deployed the 3 Swivel versions to 4 precepts with 66 total students (18, 18, 18 and 12 students) and met at either 10AM or 11AM Friday mornings. The ARCH 2112 course is a companion course to ARCH 2111 from Study F2, but students often take one or the other alone: 1/3 of the students participated in precepts from Study F2. As an incentive to return the surveys, we offered a random drawing of gift cards to students

who completed all survey materials. There were 3 instructors, one of whom taught two precepts.

At the start of the study, we collected consent forms and a demographics survey and introduced the general features of the library. We returned every two weeks to the preceptorial meetings to collect responses to the CSUQ and engagement instruments. During those times, we also announced the new Swivel version for the upcoming two weeks along with a brief description of its new features. We made those announcements when we distributed the surveys to the class at the beginning of each meeting, but the class instructors determined when students completed them (generally either at the beginning immediately after our departure or at the end of class), and collected them on behalf of us. The first survey collection was an *a priori* known exception: an exam was scheduled for the same meeting as the end of the first two-week period. Scheduling constraints prevented moving the study time frame, so students completed the survey after finishing their exam.

We balanced the order of the versions between the precepts, but since there were only 4 groups we could not perform a full counter-balance. We used 3 orderings for the 4 groups, matching the smallest group randomly with one of the other three. We used the permutations (Bar, RM, control); (RM, control, Bar); (control, bar, RM) so that every Swivel version was used by a group in each of the 3 periods.

After the last two-week period, we also gave an end-of-study questionnaire that asked students to:

- Rank the Swivel versions in order of their preference
- Describe the factors that led them to that ranking
- Rate their preference between each pair of versions on a 7-point Likert scale.

Results

Participation was a significant problem. Table 5.4 shows the response rates to each of the three surveys by precept. In Precept 2, response rates to Survey 1 were depressed compared to other sections because the two other instructors made a point of asking students to stay after the exam to finish the surveys. Response rates to Survey 2 were low because the instructor cancelled class due to illness (we collected the surveys by emailing the form to the precept and asking them to return them at the next class meeting or via email directly to us). More problematic than the overall participation rate, however was the lack of overlap in participation: because different people responded to the surveys, only 18 (27%) returned all three surveys.

But since system usage was a product of students' own volitions, a significant fraction never used the system in a given period (see Table 5.5). After combining these two facts only 6 (9%) students both returned all three surveys and used the system during each of the three survey periods. We also had no reports of any data insights, which we attribute both to relatively short usage sessions and after-the-fact self reports. However, 42 students returned at least one useful survey assessment (i.e., one from a student who used the system) over the course of our procedure (69 total useful survey assessments). Table 5.6 summarizes those assessments by interface condition (rather than time period) and dependent measure.

Table 5.4. Response rates for the three surveys by precept.

	Survey 1	Survey 2	Survey 3
Precept 1	83%	50%	42%
Precept 2	27%	33%	56%
Precept 3	72%	83%	78%
Precept 4	78%	66%	33%
Total	63%	59%	53%

Table 5.5. Frequency and median duration of use by survey period.

	Survey 1	Survey 2	Survey 3
Never	33%	51%	43%
Once or More	67%	49%	57%
Median Duration	30 min.	20 min.	20 min.

Table 5.6. Mean dependent measure values by interface condition.

	n=18 Control	n=32 Bar	n=19 RM
Enjoyment	4.57 σ=0.97	4.48 σ=1.04	4.83 σ=0.86
Engagement	4.66 σ=1.02	4.46 σ=1.40	4.34 σ=1.03
CSUQ	5.09 σ=1.16	5.05 σ=0.98	5.01 σ=0.88

Simply using a set of paired t-tests between each dependent measure allows us to include much more of our data (it excludes only those subjects who returned a single survey). Doing so inflates the type I error rate, but such pairwise comparisons result in no significant differences between any interface on any of the main dependent measures anyway. We find more interesting results conducting pairwise correlations among our dependent measures (using a Holm-Bonferroni type I error correction): CSUQ ratings of

the Bar and RM interfaces had a strong positive correlation (Pearson r^2=0.762; $p < 0.01$; n=14), and the RM and control interfaces had a strong negative correlation (Pearson r^2=-0.839; $p<0.02$; n=7). This indicates that users who liked the RM and Bar interfaces tended to like the other, and that users who liked the RM interface tended to not like the control (and *vice versa*). The Bar and control ratings were also negatively correlated, but not as strongly and not significantly (Pearson r^2=-0.513; p=0.09; n=12). From this we see that the nearly-identical mean CSUQ values in Table 5.6 mask a significant dichotomy in user ratings. The end-of-study rankings of the interfaces also display a similar pattern: the RM condition is most often ranked either first or last (see ranking histogram in Figure 5.14).

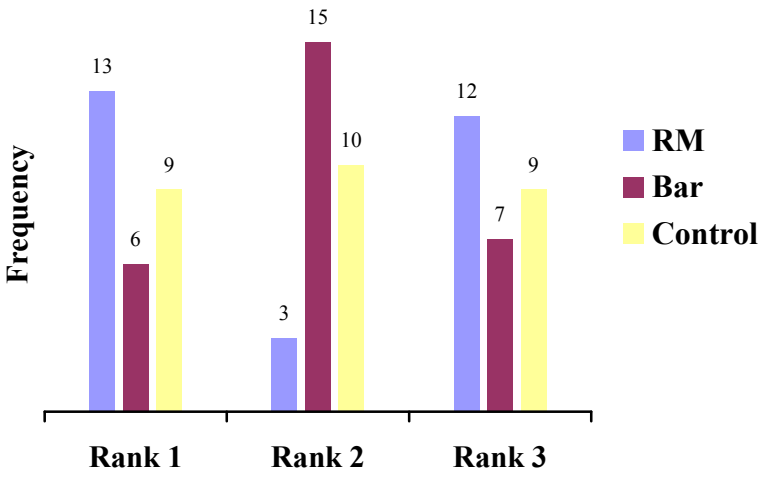

Figure 5.14. Histogram of end-of-study interface rankings.

We also conducted an analysis of the web log data collected throughout the experiment. Raw page view counts for landing (i.e., the initial library page with no filters in place) and non-landing (i.e., a page with some user-generated search or facet value

filter in place) pages broken down by filter type (search vs. facet value) are shown in Table 5.7.

Table 5.7. Landing and Non-landing (filtered) page views broken down by filter type and condition. The same non-landing page view can involve both a search term and filtering by facet value, so the total of a Facet | Search row cell may be more than the corresponding Non-Landing row cell.

	Control	Bar	RM
Landing	62	135	96
Non-Landing	65	170	138
Facet | Search	28 | 39	103 | 81	80 | 75

Several items in this table are of note. First, both the Bar and RM conditions have a smaller ratio of landing to non-landing page views, indicating that the average session consisted of more page views. Second, the ratio of facet to search filters on non-landing pages is larger for the RM and Bar conditions compared to the control: on average, those users used facet value filters more than search queries.

We also tallied the number of usage instances of the visualization components in the Bar and RM conditions. In all, users expanded 117 facet values and a ResultMap 42 times. The 42 ResultMap usages represent explicitly requested interactions, since they are hidden by default; that count is also indicative only of the initial act of showing the ResultMap, not any further interaction with it (e.g., mouseovers, etc., which we did not register in the logs). ResultMap usage was heavily biased towards their orders: the ResultMap nearest the top of the page (*Date_beg*) had 33 usages and the next-nearest (*Materials*) had the remaining 9.

Discussion

We found no evidence to support our hypotheses in general. Generally speaking, all three conditions were equivalent in the satisfaction and ranking. But our longitudinal analysis provided more significant evidence for a trend hinted at from the results of study F1: faceted ResultMap affinity appears to be bimodally distributed and is inversely related with affinity for the control non-visual condition. Our immediate theory is a relationship to individual learning style: visual learners like ResultMaps and non-visual learners find them distracting or confusing. Several free-form user responses hint at this, e.g.:

- "The Map was more of a visual aid and helped me organize my thoughts"
- "Map is the most visual and most appealing of the three. This is good, especially when dealing with architects"
- "I am a visual learner so the map version seemed to be my favorite."

But without a learning style inventory assessment this theory remains speculative.

We are surprised that the Bar interface was not better-received compared to the control. The bars were designed to be relatively unobtrusive, but some comments indicated they made the interface too distracting or dense. A few changes might mitigate such feelings: decreasing the saturation of the blue color we used; rendering the bar charts below rather than behind the text; changing sub-bars to expand on a layer above the main page so that the overall page does not move on expansion. Procedurally speaking, follow-up interviews of participants—especially those that did not like the Bar version—would have been useful to delve more into this topic.

Similarly, procedural usage prompts in the form of related class assignments would have been nice to drive additional usage of the system. Even if they were not directly targeted tasks as with study F1, students would be more likely to investigate the

image collection if its contents could be used in some kind of exercise. Such exercises in the latter two periods would have also balanced the exam event at the end of the first period. As it was, there were no significant class events other than normal lectures (apart from the exam).

CHAPTER 6:

DESIGN IMPLICATIONS FOR VISUAL SEARCH TOOLS

The final prong of our contribution is derived from our first two: given our empirical results and our theoretical contribution, what implications are there for work on future search visualization systems for digital repositories? Based on our practical experience and inferences from our faceted models, we provide an analysis and discussion of various factors affecting design success in this space and suggest some broad design recommendations. We divide these recommendations into those relating to online search visualization system design (or simply *system design* henceforth) and those relating to the design of evaluations of such systems.

6.1 System Design Implications

We frame our design implications in the context of a high-level model of system design that, given various features of a system environment, broadly suggests a system's effectiveness. In broad terms, we can consider online search UIs to consist of a base text-based component potentially augmented by visual components. Those visual augmentations can range in complexity from simple data graphics to sophisticated, dynamic infovis representations. Infovis additions have the capability of encouraging opportunistic insight but also increase the overall complexity of a search UI. Overall complexity can lead to a sort of UI overload—information overload due to the UI rather than the underlying information space. This is especially true in the DL space because so many users are first-time [39, 71]. Inherent aptitudes of users are also relevant: more

infovis augmentations are more suited to visually-oriented users. Table 6.1 summarizes these factors, their measurement scale and provides illustrative examples.

Table 6.1. Factors affecting infovis-enhanced system design effectiveness.

	Factor	Scale	Description	Illustration
User	Experience	Novice ↔ Expert	The skill level a user has with both a specific system design or with similar features from other systems.	Treemap familiarity implies more expertise with ResultMaps; usage of one faceted system implies more expertise with general faceted browsing.
User	Learning Style	Visual ↔ Non-visual	The inherent level of comfort a user has with visual representations of information.	Visual learners would be separated from non-visual (e.g., auditory, tactile, etc.) by a learning style inventory instrument.
Interface	Base Complexity	Simple ↔ Complex	The intricacy of the text-based portion of an online search system.	A keyword SERP is less complex than a faceted UI.
Interface	Visual Complexity	Simple ↔ Complex	The intricacy of the visualization portion of an online search system.	Bar charts are less complex than ResultMaps.

These factors interact in several ways:

- The overall complexity of an interface is a composition of its base and visual complexity.
- User experience is positively correlated with a higher UI overload threshold (i.e., a greater level of overall complexity).
- Overall complexity is positively correlated with opportunistic knowledge or insight discovery.
- Visual complexity has a stronger positive correlation than base complexity.

We derive several design implications from this framework. There is a tradeoff between base and visualization complexity at the margins of the UI overload: the greater the base complexity of a system, the less complex the visualization can be before users

find the UI as a whole overwhelming. User characteristics also affect this tradeoff. Users experienced with the base system are more capable of dealing with visual augmentations without overload, as are visually-oriented users.

The contrast between our SERP and faceted RM evaluations is consistent with this implication: experimental results were better when ResultMaps augmented the simpler and more familiar base interface (a SERP). Our results from Study F1 also showed a demand for simplifying the UI as a whole and the visualization specifically. Similarly, some users in F2 and F3 found the UI too 'busy' or to have 'too many links.' These kinds of comments are noticeably absent from the SERP studies, which are both simpler and more familiar to most users. Anecdotal comments from F2 and F3 are also consistent with visually-oriented users being more comfortable with visual complexity.

These relationships suggest several design consequences. Visualizations for faceted navigation are difficult since the base complexity of the system is considerable and many users are not familiar with the faceted paradigm. Our high-level model suggests that one way to incorporate faceted visualization without inducing UI overload would be to shift some of the base complexity into the visualization: for example, eliminating facet value links (reducing base complexity) and using existing ResultMap labels instead. This is also the approach taken in the FacetLens [63] and CellTrend [67] systems. However, shifting complexity to visual components may produce overload for non-visual users. An alternative is using dynamic progressive disclosure (as we did with Swivel) so that visual complexity is hidden by default, but quickly available via asynchronous requests.

From an information overload perspective, progressive disclosure has clear benefits, but other factors play into its advisability. Infovis facilitates opportunistic insight—but can do so only if the user can see infovis UI elements. Thus, there is a tension between visibility and complexity that is confounded by user characteristics. A potential improvement to simple progressive disclosure is one that factors in user experience (or proxies thereof): if the system can recognize which users have more familiarity with the system, it can change its defaults to show more complex infovis features. Similar recognition of individual visual or non-visual propensities (cf. the bimodality and negative correlations found in Study F3) could also play into that decision. Recognizing user experience specific to a particular environment impacts system design more broadly: digital repositories dominated by one-time or novice users (e.g., online public-access catalogs) are less suitable to infovis than repositories with repeated, longer term usage (e.g., data warehouses; meta-analysis of research papers).

In discussing our formal models of faceted browsing, we noted that the sizable resource costs of previews mean that—like progressive disclosure—dynamic, asynchronous is the best method for implementing such behaviors. Since previews also increase UI complexity via their interactive effects, it could be advisable to make their requests separate from normal interaction with facet values. In our implementation, hovering over a facet value link triggered a preview of that selection. While convenient, it also triggers an unnecessary preview for users who intend simply to click on the link. A small UI widget separate from the link could address this issue.

We found a strong bias in ResultMap usage towards facets that were nearest to the top of the page in Study F3. In cases when there are reliable estimates of utility for

different facets, this behavior could be beneficial: put the most useful facets with ResultMaps (or other infovis augmentation) near the top. But in general, a UI should allow facet reordering so that users who identify useful facet visualizations lower in the facet list can move them nearer the top. This also allows easier comparison of arbitrary facet visualizations.

6.2 Evaluation Design Implications

Especially given our results, the lack of statistical power inherent to the between-subjects design in most of our studies was among the most dissatisfying aspects of our evaluations. However, we are faced with a conundrum: it is difficult to separate dataset tasks from the datasets they operate on, and such tasks cannot be repeated because of practice effects. Consequently, within-subjects comparisons of interface alternatives require tasks that are different in their particulars but somehow matched. But that matching process is fraught with uncertainty—most works simply state that such tasks sets are matched along broad guidelines (e.g., lookup, complex or exploratory tasks; cf. [17, 122, 124]) with no rigorous methodology beyond intuitive comparison. Furthermore, experimental tasks are difficult to design well in the first place, especially without bias towards a particular interface—and this process requires two such sets. We faced this difficulty in Study R3, in which it was difficult to generate complex naturalistic tasks that probed ResultMaps' emphasis on outlier and cluster detection without biasing the question toward readily apparent data in the ResultMap.

Insight-based protocols [79] are an alternative to structured tasks, but seem less-well suited to DL search applications, and suffer even more from practice effects (since insight accumulates over time). Nonetheless, insight-based methods show considerable

promise, but as North notes, present considerable resource challenges. Moreover, the nature of insight itself—relatively rare, unpredictable and qualitative—makes it difficult to measure in lab studies. But evaluating insights also benefits from precision and detail about its circumstances, making it difficult to assess in longitudinal studies: diary, interviews, or other self-reported methods can be unreliable and not provide the sufficient details about insight. We certainly found this to be true in Study F3.

We already mentioned in Section 4.4 (p. 80) how we distinguish direct vs. analytical interest in a dataset. Tasks clearly influence this interest (cf. Studies R2 and F2), but inherent alignment is desirable, especially in longitudinal or open-ended studies that have no tasks (cf. Studies. R3 and F3). We also found throughout our evaluations that even loose tasks tend to generate tunnel-vision in many participants such that they are less likely to be open to opportunistic discovery. So in cases when users have an analytic interest in the data, the ability to use very open tasks also increases the chances for unplanned insights.

In accordance with this, we recommend identifying suitable dataset/user population combinations much earlier in the development process than in this work. For our evaluations, working with the curators or administrators for DL repositories for the SERP or faceted evaluations, or more senior architecture students (who might be interested in a picture of the subject matter beyond a general education requirement) might have proven more fruitful.

We suggest several other guidelines for work in this area based on this discussion. With respect to matching dataset tasks, quantitative measures of task isomorphism would be helpful. For directed tasks in hierarchical environments, such measures might include:

- Measures of similarity (via one of many metrics [14]) between the local trees around target documents.
- Measures of the attribute similarity between target items (e.g., similarity of relative distribution of differing attribute values).

For complex tasks that yield specific (or sets of) items, a *post hoc* analysis of such measures over all task results could indicate some measure of congruence. In either case, identifying high-level data similarities and crafting tasks around them might prove easier than trial-and-error task creation based on intuitive notions of similarity. This kind of analysis ignores semantic differences between information targets—which may often be more critical than the abstract structure of the information space—but any generalized cross-task comparison metric is an improvement on the status quo.

With respect to insight capture, lightweight[25] mechanisms of reporting or recording insight data are critical, especially in longitudinal studies. The exact nature of such features are dependent on circumstances, but simple data annotation tools to mark or bookmark (which can automatically trigger logging and recording features) is one possibility (*a la* sense.us [45]). Evaluation metrics impact insight reporting: using study incentives based on insight quality and frequency would certainly motivate users, especially in longitudinal studies or in cases where there is little intrinsic motivation to pursue insights about a dataset. Researchers in other fields (e.g., text input [68]) have used similar incentives with success.

Classroom evaluations can be attractive because of the ease of recruiting participants and confluences between research system and educational topics. But our experience is that students are extremely grade-driven—especially students in introductory classes. As a result, it is advisable that such evaluations have strong

[25] Lightweight refers to user rather than developer effort.

pedagogical components, especially for longitudinal assessments. Study F3 provided a useful resource for student studying for an exam in the first assessment period, but in the other two periods there was little pedagogical connection between the image library and classroom activity. The result was a dramatic decrease in usage.

6.3 Conclusions and Future Work

This work has provided three main contributions that lead to the following thesis:

> ResultMaps constitute a lightweight visualization mechanism for digital repository search systems. They provide a means for contextualizing repository content, providing several prospective benefits, while not impairing usage for uninterested users. Empirical studies of their usage, along with models of faceted environments, suggest a set of implications for design of future systems in this space and their evaluation.

We have shown that ResultMaps perform at least as well for all users over base control instances in both SERP and faceted contexts, and for some users and tasks provide significant subjective and objective benefits. We have also provided a formalism of the data and query mechanisms that underpin modern faceted UIs; those formalisms and the results of our evaluations suggest principles for designing DL search visualization systems themselves and evaluating their use.

Several avenues for future work remain both directly and indirectly related to our specific research. As we have noted, our faceted data and query models suggest several aspects of faceted UI design that might be extended in novel and useful ways (e.g., expressing uncertainty in item classification). The models also represent a way of performing a detailed complexity analysis of various faceted UI approaches by assuming a common underlying data store. As suggested in our design guidelines, developing quantitative measures for task isomorphism—even ones that ignore semantic factors—would be helpful in the evaluation of infovis or other information seeking tools.

More broadly, we see considerable similarities between the problems of evaluating information retrieval and information visualization tools in terms of task creation and validation, fruitful methods and metrics. Both areas deal with ill-formed user goals, the generic concept of data insight and difficulties matching users to appropriate datasets and tasks. Given these similarities, we see opportunity for marrying more traditional text-based IR systems with infovis techniques, especially using the guidelines we suggest here. With advancements in web technology (e.g., the HTML 5 canvas element) and the increasing integration of the web into everyday life, using the web as a platform for such integrations presents an important opportunity to distribute such integrative research into broadly-accessible tools.

A related key challenge is to integrate or reformat existing data stores for use within such novel environments: the semantic web is one such method, but in the sort term the amount of data within corporate or other proprietary organizations is problematic. We view semi-automatic tools for mapping arbitrary data into more flexible tools as a worthwhile venture: for example, a tool to analyze an existing database, suggest possible faceted data models, and construct appropriate database views—while accounting for user input and corrections at each step. Removing barriers to new data sources allows researchers in both information retrieval and information visualization to test new ideas on larger, different and more relevant data, and allows researchers to leverage existing data.

APPENDIX A:

COMBINED CSUQ AND ENGAGEMENT SURVEY INSTRUMENT

Please indicate your level of agreement with each of the following statements by circling a number from 1 to 7:

1. Overall, I am satisfied with how easy it is to use this system.

N/A Strongly Disagree 1 2 3 4 5 6 7 Strongly Agree

2. It was simple to use this system.

N/A Strongly Disagree 1 2 3 4 5 6 7 Strongly Agree

3. I can effectively complete my work using this system.

N/A Strongly Disagree 1 2 3 4 5 6 7 Strongly Agree

4. I am able to complete my work quickly using this system.

N/A Strongly Disagree 1 2 3 4 5 6 7 Strongly Agree

5. I am able to efficiently complete my work using this system.

N/A Strongly Disagree 1 2 3 4 5 6 7 Strongly Agree

6. I feel comfortable using this system.

N/A Strongly Disagree 1 2 3 4 5 6 7 Strongly Agree

7. It was easy to learn to use this system.

N/A Strongly Disagree 1 2 3 4 5 6 7 Strongly Agree

8. I believe I became productive quickly using this system.

N/A Strongly Disagree 1 2 3 4 5 6 7 Strongly Agree

9. The system gives error messages that clearly tell me how to fix problems.

N/A Strongly Disagree 1 2 3 4 5 6 7 Strongly Agree

10. Whenever I make a mistake using the system, I recover easily and quickly.

N/A Strongly Disagree 1 2 3 4 5 6 7 Strongly Agree

11. The information (online help, on-screen messages, etc) provided with this system is clear.

N/A Strongly Disagree 1 2 3 4 5 6 7 Strongly Agree

12. It is easy to find the information I needed.

N/A Strongly Disagree 1 2 3 4 5 6 7 Strongly Agree

13. The information provided for the system is easy to understand.

N/A Strongly Disagree 1 2 3 4 5 6 7 Strongly Agree

14. The information is effective in helping me complete the tasks and scenarios.

N/A Strongly Disagree 1 2 3 4 5 6 7 Strongly Agree

15. The organization of information on the system screens is clear.

N/A Strongly Disagree 1 2 3 4 5 6 7 Strongly Agree

16. The interface of this system is pleasant.

N/A Strongly Disagree 1 2 3 4 5 6 7 Strongly Agree

17. I like using the interface of this system.

N/A	Strongly Disagree	1	2	3	4	5	6	7	Strongly Agree

18. This system has all the functions and capabilities I expect it to have.

N/A	Strongly Disagree	1	2	3	4	5	6	7	Strongly Agree

19. Overall, I am satisfied with this system.

N/A	Strongly Disagree	1	2	3	4	5	6	7	Strongly Agree

*List the most **negative** aspect(s):*

_____

_____

_____

*List the most **positive** aspect(s):*

_____

_____

_____

Please check the box that best describes:

Using the system

Uninteresting	1	2	3	4	5	6	7	Interesting
Not Enjoyable	1	2	3	4	5	6	7	Enjoyable
Dull	1	2	3	4	5	6	7	Exciting
Not Fun	1	2	3	4	5	6	7	Fun

How you felt while using the retrieval system

Not absorbed intensely	1	2	3	4	5	6	7	Absorbed intensely
Attention was not focused	1	2	3	4	5	6	7	Attention was focused
Did not concentrated fully	1	2	3	4	5	6	7	Concentrated fully
Not deeply engrossed	1	2	3	4	5	6	7	Deeply engrossed

APPENDIX B:

LIBRARY KNOWLEDGE SURVEY INSTRUMENT

The contents of this library are organized into a hierarchical classification system, similar to how living things are classified (kingdom, phylum, class, order, etc.). We use the term *category* to refer to a single classification grouping. We use the term *subcategory* to refer to a category that is underneath another in the classification. Using our example, the kingdom *animalia* and the order *mammalia* are both categories, and *mammalia* is a subcategory of *animalia*.

The following questions inquire about your knowledge of the library and its classification system.

1. About how many *top-level categories* are there in the classification? _____

2. About how many *total categories* are there in the classification system? _____

3. About how many total *documents* are there in the library? _____

4. There are 11 document types in the library. The documents that appeared in the search results were most often which type (circle only one)?

Article	Lecture	Test/Exam
Audio Lecture	Reference Material	Video
Class Activity	Syllabus	Web Lecture
Homework	Tool	

5. Which of the following categories are one of the top level categories? Please circle as many as apply (0-6):

Evaluation Methods	Interface Paradigms
HCC Topics and Applications	Prototyping the UI
Information Science	Usability Principles

6. What top-level category had the ***most*** content (as measured by the number of documents in it and its subcategories)? Circle only one.

Evaluation Methods	Human Capabilities
Introduction / General	Prototyping the UI
HCC Topics and Applications	Requirements Gathering and Task Analysis
HCI Design	User Interface Software, Tools and Devices

7. What top-level category had the ***least*** content (as measured by the number of documents in it and its subcategories)? Circle only one.

Introduction / General	Evaluation Methods
Human Capabilities	HCI Design
Requirements Gathering and Task Analysis	User Interface Software, Tools and Devices
Prototyping the UI	HCC Topics and Application

APPENDIX C:

AD HOC SATISFACTION INSTRUMENT

What percentage of the tasks do you think you performed correctly?

_____

0-100%, increments of 10%

How difficult did you find these tasks?

Less difficult	1	2	3	4	5	6	7	More difficult

Did the design of the search result page make it more or less difficult to complete these tasks than a normal search result page?

Less difficult	1	2	3	4	5	6	7	More difficult
				No difference				

Did the design of the search result page make you take more or less time to complete these tasks than a normal search result page?

Less time	1	2	3	4	5	6	7	More time
				No difference				

How would you rate the search result page design as a whole?

Poor	1	2	3	4	5	6	7	Excellent

How would you rate the ***visual design*** (i.e., placement and look of the interface widgets—buttons, links, etc.—within the overall page) of the search result page?

Poor	1	2	3	4	5	6	7	Excellent

How would you rate the ***interaction design*** (i.e., capabilities and response of the interface to mouse and keyboard input) of the search result page?

Poor	1	2	3	4	5	6	7	Excellent

The following questions relate to the visual representation of the search results, which we refer to as *ResultMaps*.

On what percentage of the tasks did you use the ResultMap display?

_____

0-100%, increments of 10%

On what percentage of the tasks that you used the ResultMaps did the ResultMaps help you complete the task?

_____

0-100%, increments of 10%

[If ResultMaps helped you]
What aspect(s) of ResultsMaps helped you complete the tasks?

Grouping search results into categories:

Not Helpful	1	2	3	4	5	6	7	Very Helpful

Coloring search results by document type:

Not Helpful	1	2	3	4	5	6	7	Very Helpful

Other (please specify): _____

Not Helpful	1	2	3	4	5	6	7	Very Helpful

What visually didn't you like about the ResultMap display?

What functionally (e.g., action-oriented effects such as those from clicking, mouse movements, etc.) didn't you like about the ResultMap display?

What suggestions do you have for making the ResultMaps more useful?

REFERENCES

[1] *Apple Human Interface Guidelines*. Available from: http://developer.apple.com/documentation/userexperience/Conceptual/AppleHIGuidelines/ (accessed May 2009).

[2] *Longwell RDF Browser*. Available from: http://simile.mit.edu/longwell/ (accessed May 2009).

[3] Ahlberg, C., Williamson, C., and Shneiderman, B. Dynamic Queries for Information Exploration: An Implementation and Evaluation. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 1992*, pp. 619-626.

[4] Allen, R. Retrieval from Facet Spaces, *Electronic Publishing* 8(2 and 3), pp. 247-257.

[5] Amar, R., Eagan, J., and Stasko, J. Low-Level Components of Analytic Activity in Information Visualization. In *Proceedings of the IEEE Symposium on Information Visualization 2005*, pp. 111-117.

[6] Anderson, C. Record Relationship Navigation: Implications for Information Access and Discovery. In Proceedings of the HCIR Workshop 2007, p. 6.

[7] Andrews, K., Gutl, C., Moser, J., Sabol, V., and Lackner, W. Search Result Visualisation with xFIND. In Proceedings of the IEEE Workshop on User Interfaces to Data Intensive Systems 2001, pp. 50-58.

[8] Andrews, K. and Heidegger, H. Information Slices: Visualising and Exploring Large Hierarchies Using Cascading, Semi-Circular Discs. In *Proceedings of the IEEE Symposium on Information Visualization Late Breaking Hot Topics 1998*, pp. 9-12.

[9] Balzer, M. and Deussen, O. Voronoi Treemaps. In *Proceedings of the IEEE Symposium on Information Visualization 2005*, pp. 49-56.

[10] Bederson, B. Interfaces for Staying in the Flow, *ACM Ubiquity* 5(27).

[11] Bederson, B.B., Shneiderman, B., and Wattenberg, M. Ordered and Quantum Treemaps: Making Effective Use of 2D Space to Display Hierarchies, *ACM Trans. Graph.* 21(4), pp. 833-854.

[12] Bendix, F., Kosara, R., and Hauser, H. Parallel Sets: Visual Analysis of Categorical Data. In *Proceedings of the IEEE Symposium on Information Visualization 2005*, pp. 133-140.

[13] Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., and Sheets, D. Tabulator: Exploring and Analyzing Linked Data on the Semantic Web. In Proceedings of the International Semantic Web User Interaction Workshop 2006.

[14] Bille, P. A Survey on Tree Edit Distance and Related Problems, *Theor. Comput. Sci.* 337(1-3), pp. 217-239.

[15] Börner, K., Dillon, A., and Dolinsky, M. Lvis-Digital Library Visualizer. In *Proceedings of the IEEE International Conference on Information Visualization 2000*, pp. 77-81.

[16] Bruls, M., Huizing, K., and Wijk, J.V. Squarified Treemaps. In *Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization 2000*, pp. 33-42.

[17] Capra, R., Marchionini, G., Oh, J.S., Stutzman, F., and Zhang, Y. Effects of Structure and Interaction Style on Distinct Search Tasks. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries 2007*, pp. 442-451.

[18] Card, S., Mackinlay, J., and Shneiderman, B. *Readings in Information Visualization: Using Visualization to Think*, Morgan Kaufmann: 1999.

[19] Chen, H., Houston, A., Sewell, R., and Schatz, B. Internet Browsing and Searching: User Evaluations of Category Map and Concept Space Techniques, *Journal of the American Society for Information Science* 49(7), pp. 582-603.

[20] Chen, P. The Entity-Relationship Model: Toward a Unified View of Data, *ACM Trans. Database Syst.* 1(1), pp. 9-36.

[21] Chin, J., Diehl, V., and Norman, K. Development of an Instrument Measuring User Satisfaction of the Human-Computer Interface. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 1988*, pp. 213-218.

[22] Chong, E., Das, S., Eadon, G., and Srinivasan, J. An Efficient SQL-Based RDF Querying Scheme. In *Proceedings of the ACM International Conference on Very Large Databases 2005*, pp. 1216-1227.

[23] Chuah, M. Dynamic Aggregation with Circular Visual Designs. In *Proceedings of the IEEE Symposium on Information Visualization 1998*, pp. 35-43.

[24] Clarkson, E., Day, J., and Foley, J. The Development of an Educational Digital Library for Human-Centered Computing. GVU Tech Report GIT-GVU-05-33.

[25] Clarkson, E., Day, J., and Foley, J. An Educational Digital Library for Human-Centered Computing. In *Proceedings of the ACM Conference on Human Factors in Computing Extended Abstracts 2006*, pp. 646-651.

[26] Clarkson, E., Foley, J.D., and Desai, K. ResultMaps: Visualization for Search Interfaces, *IEEE Transactions on Visualization and Computer Graphics*, to appear.

[27] Clarkson, E., Navathe, S., and Foley, J. Generalized Formal Models for Faceted User Interfaces. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries 2009*, to appear.

[28] Dachselt, R., Frisch, M., and Weiland, M. FacetZoom: A Continuous Multi-Scale Widget for Navigating Hierarchical Metadata. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 2008*, pp. 1353-1356.

[29] Derthick, M. *Bungee View*. Available from: http://cityscape.inf.cs.cmu.edu/bungee/ (accessed May 2009).

[30] Elmasri, R. and Navathe, S. *Fundamentals of Database Systems* 5th ed., Addison Wesley: 2007.

[31] Fekete, J.-D., Wijk, J.J., Stasko, J.T., and North, C. "The Value of Information Visualization", in *Information Visualization: Human-Centered Issues and Perspectives*. Springer-Verlag, 2008. pp. 1-18.

[32] Fekete, J.D. and Plaisant, C. Interactive Information Visualization of a Million Items. In *Proceedings of the IEEE Symposium on Information Visualization 2002*, pp. 117-124.

[33] Foley, J., Beaudouin-Lafon, M., Grudin, J., Hollan, J., Hudson, S., Olson, J., and Verplank, B. Graduate Education in Human-Computer Interaction. In *Proceedings of the ACM Conference on Human Factors in Computing Systems Extended Abstracts 2005*, pp. 2113-2114.

[34] Furnas, G. Generalized Fisheye Views. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 1986*, pp. 16-23.

[35] Ghani, J.A., Supnick, R., and Rooney, P. The Experience of Flow in Computer-Mediated and in Face-to-Face Groups. In *Proceedings of International Conference on Information Systems 1991*, pp. 229-237.

[36] Good, L., Popat, A., Janssen, W., and Bier, E. A Fluid Treemap Interface for Personal Digital Libraries. In *Proceedings of European Conference on Research and Advanced Technology for Digital Libraries 2005*, pp. 162-173.

[37] Gray, W.D., John, B.E., and Atwood, M.E. The Precis of Project Ernestine or an Overview of a Validation of GOMS. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 1992*, pp. 307-312.

[38] Guan, Z. and Cutrell, E. An Eye Tracking Study of the Effect of Target Rank on Web Search. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 2007*, pp. 417-420.

[39] Harley, D. and Henke, J. Toward an Effective Understanding of Website Users: Advantages and Pitfalls of Linking Transaction Log Analyses and Online Surveys. *D-Lib Magazine* **13**(3/4). 2007.

[40] Hart, S.G. and Stavelan, L.E. Development of Nasa-Tlx (Task Load Index): Results of Empirical and Theoretical Research, *Human Mental Workload*, pp. 139-183.

[41] Hearst, M. Tilebars: Visualization of Term Distribution Information in Full Text Information Access. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 1995*, pp. 59-66.

[42] Hearst, M., Elliott, A., English, J., Sinha, R., Swearingen, K., and Yee, K.-P. Finding the Flow in Web Site Search, *Commun. ACM* 45(9), pp. 42-49.

[43] Hearst, M. and Karadi, C. Cat-a-Cone: An Interactive Interface for Specifying Searches and Viewing Retrieval Results Using a Large Category Hierarchy. In *Proceedings of the ACM Conference on Information Retrieval 1997*, pp. 246-255.

[44] Heer, J., Card, S., and Landay, J. Prefuse: A Toolkit for Interactive Information Visualization. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 2005*, pp. 421-430.

[45] Heer, J., Viégas, F., and Wattenberg, M. Voyagers and Voyeurs: Supporting Asynchronous Collaborative Information Visualization. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 2007*, pp. 1029 - 1038.

[46] Huynh, D. *The Nested Faceted Browser*. Available from: http://people.csail.mit.edu/dfhuynh/projects/nfb/ (accessed May 2009).

[47] Huynh, D. *Parallax*. Available from: http://mqlx.com/~david/parallax/ (accessed May 2009).

[48] Inselberg, A. The Plane with Parallel Coordinates, *The Visual Computer* 1(4), pp. 69-91.

[49] Jerding, D.F. and Stasko, J. The Information Mural: A Technique for Displaying and Navigating Large Information Spaces. In *Proceedings of the IEEE Symposium on Information Visualization 1995*, pp. 43-50.

[50] John, B.E. and Kieras, D.E. The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast, *ACM Trans. Comput.-Hum. Interact.* 3(4), pp. 320-351.

[51] John, B.E., Prevas, K., Salvucci, D.D., and Koedinger, K. Predictive Human Performance Modeling Made Easy. In Proceedings of the ACM Workshop on Proceedings of the SIGCHI conference on Human factors in computing systems 2004.

[52] Johnson, B. and Shneiderman, B. Tree-Maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures. In *Proceedings of the IEEE Conference on Visualization 1991*, pp. 284-291.

[53] Johnston, R. *Analytic Culture in the United States Intelligence Community: An Ethnographic Study*, Central Intelligence Agency: 2005.

[54] Käki, M. Findex: Search Result Categories Help Users When Document Ranking Fails. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 2005*, pp. 131-140.

[55] Kampanya, N., Shen, R., Kim, S., North, C., and Fox, E. Citiviz: A Visual User Interface to the CITIDEL System. In *Proceedings of European Conference on Digital Libraries 2004*, pp. 12-17.

[56] Kang, S., Pourang, I., and Li, B. An Evaluation of Content Browsing Techniques for Hierarchical Space-Filling Visualizations. In *Proceedings of the IEEE Symposium on Information Visualization 2005*, pp. 81-88.

[57] Klerkx, J., Duval, E., and Meire, M. Using Information Visualization for Accessing Learning Object Repositories. In *Proceedings of the IEEE International Conference on Information Visualisation 2004*, pp. 465-470.

[58] Kobilarov, G. and Dickinson, I. Humboldt: Exploring Linked Data. In Proceedings of the WWW Workshop on Linked Data on the Web 2008.

[59] Kobsa, A. User Experiments with Tree Visualization Systems. In *Proceedings of the IEEE Symposium on Information Visualization 2004*, pp. 9-16.

[60] Kohonen, T. The Self-Organizing Map, *Proceedings of the IEEE* 78(9), pp. 1464-1480.

[61] Kules, W. *Supporting Exploratory Web Search with Meaningful and Stable Categorized Overviews*. Ph.D. Dissertation, University of Maryland. 2006, HCIL-2006-14.

[62] Lamping, J., Rao, R., and Pirolli, P. A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 1995*, pp. 401-408.

[63] Lee, B., Smith, G., Robertson, G., Czerwinski, M., and Tan, D. FacetLens: Exposing Trends and Relationships to Support Sensemaking within Faceted

Datasets. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 2009*, p. to appear.

[64] Levene, M. and Loizou, G. Why Is the Snowflake Schema a Good Data Warehouse Design?, *Information Systems* 28(3), pp. 225-240.

[65] Lewis, J.R. IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use, *Int. J. Hum.-Comput. Interact.* 7(1), pp. 57-78.

[66] Lin, X. Map Displays for Information Retrieval, *J. Am. Soc. Inf. Sci.* 48(1), pp. 40-54.

[67] Liu, Z., Stasko, J., and Sullivan, T. Celltrend: Inter-Attribute Visual Analysis of Temporal Transaction Data. In *Proceedings of the IEEE Conference on Information Visualization 2009*, p. in submission.

[68] Lyons, K., Starner, T., and Gane, B. Experimental Evaluations of the Twiddler One-Handed Chording Mobile Keyboard, *Hum.-Comput. Interact.* 21(4), pp. 343-392.

[69] MacKenzie, I.S. and Buxton, W. Extending Fitts' Law to Two-Dimensional Tasks. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 1992*, pp. 219-226.

[70] Malinowski, E. and Zimányi, E. Hierarchies in a Multidimensional Model: From Conceptual Modeling to Logical Representation, *Data & Knowledge Engineering* 59(2), pp. 348-377.

[71] Marchionini, G. Evaluating Digital Libraries: A Longitudinal and Multifaceted View, *Library Trends* 49(2), pp. 304-333.

[72] Marchionini, G. and Brunk, B. Toward a General Relation Browser: A GUI for Information Architects, *Journal of Digital Information* 4(1).

[73] Marchionini, G. and Shneiderman, B. Finding Facts Vs. Browsing Knowledge in Hypertext Systems, *IEEE Computer* 21(1), pp. 70-80.

[74] McGuffin, M.J. and schraefel, m.c. A Comparison of Hyperstructures: Zzstructures, Mspaces, and Polyarchies. In *Proceedings of the ACM Conference on Hypertext and Hypermedia 2004*, pp. 153-162.

[75] Melton, J. *Understanding the New SQL: A Complete Guide*. Vol. I, 2nd ed., Morgan Kaufmann: 2000.

[76] Nielsen, J. Mental Models for Search Are Getting Firmer. *Alertbox Column*, May 9, 2005. Available from: http://www.useit.com/alertbox/20050509.html.

[77] Nielsen, J. Scrolling and Scrollbars. *Alertbox Column*, July 11, 2005. Available from: http://www.useit.com/alertbox/20050711.html.

[78] Nielsen, J. and Loranger, H. *Prioritizing Web Usability*, New Riders: 2006.

[79] North, C. Toward Measuring Visualization Insight, *IEEE Comput. Graph. Appl.* 26(3), pp. 6-9.

[80] Nowell, L., France, R., Hix, D., Heath, L., and Fox, E. Visualizing Search Results: Some Alternatives to Query-Document Similarity. In *Proceedings of the ACM Conference on Information Retrieval 1996*, pp. 67-75.

[81] Olsen, K., Korfhage, R., Sochats, K., Spring, M., and Williams, J. Visualization of a Document Collection: The VIBE System, *Information Processing and Management* 29(1), pp. 69-81.

[82] Olston, C. and Chi, E.H. ScentTrails: Integrating Browsing and Searching on the Web, *ACM Trans. Comput.-Hum. Interact.* 10(3), pp. 177-197.

[83] Oren, E., Heitmann, B., and Decker, S. Extending Faceted Navigation for RDF Data. In *Proceedings of International Semantic Web Conference 2006*, pp. 559-572.

[84] Pirolli, P. Computational Models of Information Scent-Following in a Very Large Browsable Text Collection. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 1997*, pp. 3-10.

[85] Pirolli, P. and Card, S. Information Foraging, *Psychological Review* 106(4), pp. 643-675.

[86] Plaisant, C. The Challenge of Information Visualization Evaluation. In *Proceedings of the ACM Conference on Advanced Visual Interfaces 2004*, pp. 109-116.

[87] Plaisant, C., Fekete, J.D., and Grinstein, G. Promoting Insight-Based Evaluation of Visualizations: From Contest to Benchmark Repository, *Visualization and Computer Graphics, IEEE Transactions on* 14(1), pp. 120-134.

[88] Plaisant, C., Grosjean, J., and Bederson, B. Spacetree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation. In *Proceedings of the IEEE Symposium on Information Visualization 2002*, pp. 57-64.

[89] Plaisant, C., Shneiderman, B., Doan, K., and Bruns, T. Interface and Data Architecture for Query Preview in Networked Information Systems, *ACM Trans. Inf. Syst.* 17(3), pp. 320-341.

[90] Pollitt, A. The Key Role of Classification and Indexing in View-Based Searching. Presented at 63rd IFLA General Conference 1997. Available from: http://www.ifla.org/IV/ifla63/63polst.pdf.

[91] Pousman, Z., Romero, M., Smith, A., and Mateas, M. Living with Tableau Machine: A Longitudinal Investigation of a Curious Domestic Intelligence. In *Proceedings of the ACM Conference on Ubiquitous computing 2008*, pp. 370-379.

[92] Ranganathan, S. *Elements of Library Classification*, Asia Publishing House: 1962.

[93] Robertson, G., Cameron, K., Czerwinski, M., and Robbins, D. Polyarchy Visualization: Visualizing Multiple Intersecting Hierarchies. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 2002*, pp. 423-430.

[94] Robertson, G.G., Mackinlay, J.D., and Card, S.K. Cone Trees: Animated 3D Visualizations of Hierarchical Information. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 1991*.

[95] Rohrer, R.M. and Swing, E. Web-Based Information Visualization, *IEEE Computer Graphics and Applications* 17(4), pp. 52-59.

[96] Salampasis, M. and Diamantaras, K. Experimental User-Centered Evaluation of an Open Hypermedia System and Web Information Seeking Environments, *Journal of Digital Information* 2(4).

[97] Saracevic, T. Digital Library Evaluation: Toward Evolution of Concepts, *Library Trends* 49(2), pp. 350-369.

[98] Saraiya, P., North, C., and Duca, K. An Insight-Based Methodology for Evaluating Bioinformatics Visualizations, *IEEE Transactions on Visualization and Computer Graphics* 11(4), pp. 443-456.

[99] Saraiya, P., North, C., Lam, V., and Duca, K.A. An Insight-Based Longitudinal Study of Visual Analytics, *IEEE Transactions on Visualization and Computer Graphics* 12(6), pp. 1511-1522.

[100] schraefel, m.c., Smith, D.A., Owens, A., Russell, A., Harris, C., and Wilson, M. The Evolving mSpace Platform: Leveraging the Semantic Web on the Trail of the Memex. In *Proceedings of the ACM Conference on Hypertext and Hypermedia 2005*, pp. 174-183.

[101] Sebrechts, M.M., Cugini, J.V., Laskowski, S.J., Vasilakis, J., and Miller, M.S. Visualization of Search Results: A Comparative Evaluation of Text, 2D, and 3D Interfaces. In *Proceedings of the ACM Conference on Information Retrieval 1999*, pp. 3-10.

[102] Shen, R., Vemuri, N.S., Fan, W., Torres, R.d.S., and Fox, E.A. Exploring Digital Libraries: Integrating Browsing, Searching, and Visualization. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries 2006*, pp. 1-10.

[103] Shneiderman, B., Feldman, D., Rose, A., and Grau, X. Visualizing Digital Library Search Results with Categorical and Hierarchical Axes. In *Proceedings of the ACM Conference on Digital Libraries 2000*, pp. 57-66.

[104] Smith, G., Czerwinski, M., Meyers, B.R., Robertson, G., and Tan, D.S. Facetmap: A Scalable Search and Browse Visualization, *IEEE Transactions on Visualization and Computer Graphics* 12(5), pp. 797-804.

[105] Soukoreff, R.W. and MacKenzie, I.S. Measuring Errors in Text Entry Tasks: An Application of the Levenshtein String Distance Statistic. In *Proceedings of the ACM Conference on Human Factors in Computing Systems Extended Abstracts 2001*, pp. 319-320.

[106] Stasko, J., Catrambone, R., Guzdial, M., and McDonald, K. An Evaluation of Space-Filling Information Visualizations for Depicting Hierarchical Structures, *Int. J. Hum.-Comput. Stud.* 53(5), pp. 663-694.

[107] Stasko, J., McColgin, D., Miller, T., Plaue, C., and Pousman, Z. Evaluating the Infocanvas Peripheral Awareness System: A Longitudinal, in-Situ Study. GVU Tech Report GIT-GVU-05-08. 2005.

[108] Stefaner, M., Urban, T., and Seefelder, M. Elastic Lists for Facet Browsing and Resource Analysis in the Enterprise. In Proceedings of the IEEE Workshop on FIND at DEXA 2008.

[109] Sumner, T. and Dawe, M. Looking at Digital Library Usability from a Reuse Perspective. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries 2001*, pp. 416-425.

[110] Sumner, T. and Marlino, M. Digital Libraries and Educational Practice: A Case for New Models. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries 2004*, pp. 170-178.

[111] Tweedie, L., Spence, B., Williams, D., and Bhogal, R. The Attribute Explorer. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 1994*, pp. 435-436.

[112] Veerasamy, A. and Belkin, N.J. Evaluation of a Tool for Visualization of Information Retrieval Results. In *Proceedings of the ACM Conference on Information Retrieval 1996*, pp. 85-92.

[113] Viegas, F.B., Wattenberg, M., McKeon, M., Ham, F.v., and Kriss, J. Harry Potter and the Meat-Filled Freezer: A Case Study of Spontaneous Usage of

Visualization Tools. In *Proceedings of the Hawaii International Conference on System Sciences 2008*, pp. 159-168.

[114] Vliegen, R., van Wijk, J.J., and van der Linden, E.J. Visualizing Business Data with Generalized Treemaps, *IEEE Transactions on Visualization and Computer Graphics* 12(5), pp. 789-796.

[115] Wattenberg, M. Visualizing the Stock Market. In *Proceedings of the ACM Human Factors in Computing Systems Extended Abstracts 1999*, pp. 188-189.

[116] Wehrend, S. and Lewis, C. A Problem-Oriented Classification of Visualization Techniques. In *Proceedings of the IEEE Conference on Visualization 1990*, pp. 139-143.

[117] Willett, W., Heer, J., and Agrawala, M. Scented Widgets: Improving Navigation Cues with Embedded Visualizations, *Visualization and Computer Graphics, IEEE Transactions on* 13(6), pp. 1129-1136.

[118] Williamson, C. and Shneiderman, B. The Dynamic HomeFinder: Evaluating Dynamic Queries in a Real-Estate Information Exploration System. In *Proceedings of the ACM Conference on Information Retrieval 1992*, pp. 338-346.

[119] Wilson, M.L., Andre, P., and schraefel, m.c. Backward Highlighting: Enhancing Faceted Search. In *Proceedings of the ACM Symposium on User Interface Software and Technology 2008*, pp. 235-238.

[120] Wise, J.A., Thomas, J.J., Pennock, K., Lantrip, D., Pottier, M., Schur, A., and Crow, V. Visualizing the Non-Visual: Spatial Analysis and Interaction with Information from Text Documents. In *Proceedings of Information Visualization 1995*, pp. 51-58.

[121] Xie, H. Shifts of Interactive Intentions and Information-Seeking Strategies in Interactive Information Retrieval, *Journal of the American Society for Information Science* 51(9), pp. 841-857.

[122] Yee, K.-P., Swearingen, K., Li, K., and Hearst, M. Faceted Metadata for Image Search and Browsing. In *Proceedings of the ACM Conference on Human Factors in Computing Systems 2003*, pp. 401-408.

[123] Yi, J.S., Kang, Y.a., Stasko, J., and Jacko, J. Toward a Deeper Understanding of the Role of Interaction in Information Visualization, *IEEE Transactions on Visualization and Computer Graphics* 13(6), pp. 1224-1231.

[124] Zhang, J. and Marchionini, G. Evaluation and Evolution of a Browse and Search Interface: Relation Browser++. In *Proceedings of the National Conference on Digital Government Research 2005*, pp. 179-188.

[125] Zloof, M. Query-by-Example: A Data Base Language, *IBM Systems Journal* 16(4), pp. 324-343.